Bandit Algorithms for Recommender Systems

by

Zeynep Melda Korkut

Submitted to the Tepper School of Business
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

CARNEGIE MELLON UNIVERSITY

May 2022

Committee Members
Andrew A. Li
David Choi
Zack Lipton
Ben Moseley

Bandit Algorithms for Recommender Systems

by

Zeynep Melda Korkut

Submitted to the Tepper School of Business
on May 15, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The multi-armed bandit is by now the de facto model for exploration vs. exploitation problems arising in recommender systems. Yet, there still exists a wide gap between the bandit models studied in theory and the problems faced in the real world. The purpose of this dissertation is to bridge this gap.

In Chapter 1, we study the classic stochastic linear bandit problem under the restriction that each arm may be selected for limited number of times. This simple constraint, which we call disposability, captures a common restriction that occurs in recommendation problems from a diverse array of applications ranging from personalized styling services to dating platforms. We show that the regret for this problem is characterized by a previously-unstudied function of the reward distribution among optimal arms. Algorithmically, our upper bound relies on an optimism-based policy which, while computationally intractable, lends itself to approximation via a fast alternating heuristic initialized with a classic similarity score. Experiments show that our policy dominates a set of benchmarks which includes algorithms known to be optimal for the linear bandit without disposability, along with natural modifications to these algorithms for the disposable setting.

In Chapter 2, we introduce a new notion of meta-learning in the linear bandit setting: a sequence of N 'episodes' is played, each episode consisting of a linear bandit played for T periods with a separate unknown parameter vector in dimension d. Distinct from previous work, the unknown vectors across the episodes are assumed to lie in a (unknown) m-dimensional linear subspace, (where typically $s \ll T \ll d$). The notion of meta-learning then corresponds to learning this subspace, and meta-regret is measured with respect to the optimal achievable reward when this subspace (but not the exact episodic vectors) is known. This setting subsumes the usual meta-learning analogue of the standard finite-armed bandit, and naturally models applications such as the two-sided cold start problem in recommender systems. We propose an algorithm for this problem which achieves $\tilde{O}(d + s\sqrt{dNT})$ meta-regret, which we show is order-optimal in N, T, and d by proving a corresponding lower bound of $\Omega(d + \sqrt{sdNT} + Ns\sqrt{T})$. Surprisingly, our algorithm is entirely greedy (i.e. makes no attempt to explore) at the meta-learning level, consisting of a classic optimism-based algorithm applied to projections onto greedily-selected s-dimensional subspaces. Experiments show that our algorithm significantly outperforms natural benchmark algorithms.

In Chapter 3, we further generalize the meta-linear bandit to allow for arbitrary user arrivals. We propose an algorithm that recovers the underlying lower-dimensional subspace by solving a convex relaxation of a novel rank-minimization problem, followed by an optimism-based decision that leverages recent results in uncertainty quantification for matrix completion algorithms. We test our algorithm against various benchmarks from the literature on real-world data from NetEase, a music streaming platform, finding that our algorithm dominates all benchmarks.

Acknowledgments

It is quite difficult to acknowledge every person that has supported me in some way throughout this long and rewarding journey. Nevertheless, here is my best attempt.

I would like to thank first and foremost Andrew Li, who has given me invaluable advice for every aspect of life, for working with me throughout my journey, for supporting me academically and helping me get the experience I needed. This would not be possible without the interesting problems he wanted to work on and the enlightening discussions we had. I also want to thank the rest of my committee members Zack Lipton, Ben Moseley, and Dave Choi for the efforts they put and the time they spent. I want to especially thank Zack Lipton for the interesting and challenging discussions he always brings to the table, being enthusiastic about work, and being the friendly face on our floor. I want to further express my appreciation to John Hooker, who has provided me with the opportunity to work on unorthodox and interesting projects, given me invaluable advice, and engaged in intellectually fulfilling conversations with me.

I would like to further thank my cohort, Ozgun Elci, Sagnik Das, Yuyan Wang, Violet Chen and Thomas Lavastida for working with me on the various challenging problems especially during the first couple of years in this program, and for the fun memories we shared, the intellectual discussions and true companionship. It is practically impossible to learn so much without some help from others, each and every one in the program has taught me so much. I would like to especially thank my roommate of many years, Musa Celdir, with whom I shared a great deal of life experience and friendship, for giving any kind of support in challenging times, being there no matter what, and being a brother that everyone should have. I want to further express my gratitude to Laila Lee and Lawrence Rapp, for accommodating every need we had, for being so supportive, friendly and bringing joy to any event and encounter we had.

Last but not least, I am inexplicably grateful to my parents and my sister. They supported me through every step of my academic life for as long as I remember, they encouraged me, and always believed in me even when I had doubts.

Contents

Chapter 1

Introduction

During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries). Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors.

Aside from the real world applications, recommender system algorithms have found a solid voice among academics, especially in machine learning community. Bandit algorithms and matrix completion methods take the lead when it comes to the most popular methods developed to solve these problems. For a survey on these methods, see the following surveys: [69], [15], [62].

This thesis focuses on a particular issue faced within academia and machine learning practitioners: despite the abundance of work done to perfect recommender systems, there still exists a wide gap between the bandit models studied in theory and the problems faced in the real world. We, hereby, introduce 3 different settings where we give algorithms to solve each issue faced by practitioners from an academic point of view.

1.1 Chapter 1: Disposable Linear Bandits

Consider the stochastic linear bandit problem: given a set of 'arms' $a_1, \ldots, a_k \in \mathbb{R}^d$, the task is to select a single arm in each of a sequence of time periods so as to maximize the total reward gained. The rewards are random, but there exists some (unknown) $\theta^* \in \mathbb{R}^d$ such that the mean reward when an arm a_i is chosen is equal to $\langle a_i, \theta^* \rangle$. Algorithms known to be theoretically optimal (in a sense we will review later on) for this problem successfully balance the trade-off between arms that are known to yield high reward, with arms that may reveal more information about the unknown θ^*.

These algorithms are now a core component of most modern recommender systems, as the linear bandit model provides an extremely close fit to a common problem in recommender systems: the *cold-start* problem. Succinctly, this is the problem of making recommendations to newer users whose amount of activity is insufficient to accurately estimate their preferences. To map the cold-start problem to the linear bandit model: the a_i's encode the 'features' for each item, which have been previously estimated using past data, and θ^* encodes the unknown features of the new user. The linear form of the mean reward matches the underlying preference model of a variety of recommendation algorithms including those based on matrix factorization.

The key problem we seek to address is for a particular subset of recommender systems in which the number of recommendations of an item made to a person *may not exceed a certain limit*, a property we will refer to as *disposability*. One natural example of disposability in practice is personalized styling services (e.g. Stitch Fix, BeautyFIX, Trunk Club) which by and large operate by sending personalized 'boxes' of items to users. Users can choose to purchase any subset of the box, and these companies do not re-send an item to the same user.

Thus motivated, we study the linear bandit problem under the additional rule that each arm may be selected at most a certain amount for each user. Our primary observation is that this disposability constraint impacts both (a) the performance of algorithms known to be optimal when disposability is not imposed, and (b) the nature of algorithms that achieve meaningful theoretical (regret) guarantees. Our specific contributions are as follows:

1. We prove a minimax lower bound of $\Omega(\phi d\sqrt{T})$ on the regret of the disposable linear

bandit problem which suggests that disposability should allow for lower regret than the standard linear bandit. In particular, while the dependence on the dimensionality d and time horizon T remains the same as in the standard linear bandit, a new term $\phi \in [0, 1]$ loosely captures the 'spread' of the optimal set of arms.

2. We propose an optimism-based algorithm, which generalizes the optimal upper confidence bound algorithm (called *LinUCB*) for the standard linear bandit. We prove an upper bound of $O(\sqrt{\overline{\gamma\rho}}d\sqrt{T})$ on its regret, where γ and ρ are parameters that again encodes a notion of the spread of the optimal arm set. Noting that our algorithm involves a computationally hard optimization formulation, we propose a fast alternating heuristic. While this heuristic is not guaranteed to find the true optimum, we observe that it performs well in practice when combined with a clever initialization based on similarity scores.

3. We evaluate our algorithm's performance on a recommendation task based on data from a large online dating platform. Compared to a number of benchmarks, including LinUCB and a natural modification of Thompson sampling, our algorithm (solved via the preceding heuristic) achieves as much as 10% lower regret against all competitors.

1.2 Chapter 2: Meta Linear Bandits

Stochastic linear bandits is an online learning model that associates each action with a feature vector and assumes the mean reward is the inner product between the feature vector and an unknown parameter vector [1], [24], [25]. There are numerous applications that make use of this framework, such as serial webpage recommendations, pricing, or various other problems about modeling user behavior. Generally speaking, the idea is to learn these unknown parameters of users within a time frame called *horizon*.

Previous work on linear bandits has focused primarily on the case where the horizon is much larger than the dimension of the space. The main reason for this is that the minimax lower bounds are linear as opposed to sublinear in horizon in the worst case when the opposite is true, and the desired rate of learning may not happen in an adversarial setting. However,

many real world applications deal with high dimensional settings in which the horizon may not be as large as these algorithms require. The framework we propose is what sheds some light into this problem: we can learn from the combination of different games where the dimension is much larger and the underlying structure of these distinct parameters is low-rank. In other words, on a higher level we also learn about the lower dimensional space from which the parameters come from. We place this idea in the meta learning with bandits literature.

Consider an online system where we see a series of users coming in, staying for a fixed amount of rounds, receiving recommendations, observing rewards and leaving. The platform can consider the rewards as implicit, such as the time spent, or explicit such as the binary 'click-no click' proxy measure. The problem the platform is concerned with is that, given that these users have unknown feature vectors coming from a lower dimensional subspace, what kind of approach can we take to learn more and/or better about the current and future users so as to maximize the cumulative reward? The problem reveals itself to be 2-fold: the individual parameter learning part, which is what most of bandit literature is focused on, and the meta learning part on a higher layer, which is learning the underlying structure these parameters come from.

Paper	Bandit Type	Meta Regime	Assumptions
[39]	Linear bandits	Sparsity	High dimensional
[21]	Linear Bandits	Bias	Common distribution
[8]	Linear Bandits	Distribution prior	Knowledge of Prior
This paper	Linear Bandits	Low dimensional subspace	High dimensional

Meta bandits are a part of meta learning, the 'learning-to-learn' setting where the agent has extra information on the structure behind the incoming parameters to the system and learns the parameters to that of the structure as well. In the bandit world, the algorithms also try to learn the underlying information about these parameters so as to use it to discover the parameters themselves. We see different regimes of meta learning within bandits literature such as models with a prior knowledge on the distribution over these parameters [21], or over the arms [49], meta learning over different bandit models [83], sparsity [39], low rank reward matrix where the parameters are tensors [57]. We give a more structured explanation in the above table. Our work fills the gap where the parameters come from an unknown lower

dimensional subspace and the dimension we operate on can be much larger than the horizon.

In meta learning, we also define an *oracle* with the knowledge of this underlying structure. For instance, if these parameters come from a common distribution, then the oracle has the exact knowledge on the parameters of this distribution. In a sparse setting, it is the knowledge on which dimensions are active dimensions. In our case, it is the knowledge of the lower dimensional space, or more formally, the knowledge of the spanning vectors which span the $s-$dimensional linear subspace. The meta level algorithm performances are compared to that of *oracle* which acts as the baseline.

Our Contributions Our work fills in the gap in the literature where we see a high dimensional multi-episode setting with a meta level regime of low rank design. We highlight the main contributions below.

1. Algorithm: Having introduced the new concept of meta linear bandits in high dimensional regime, we propose an algorithm Projected LinUCB that relies on *greedy* projections on the meta level, and lower dimensional optimistic UCB-based algorithms on the episodic level.

2. Theory: We give a lower bound of $\Omega(d + \sqrt{sdNT} + Ns\sqrt{T})$ based on a special case: sparse linear bandits. We further give upper bounds on the algorithm $\tilde{O}(d + s\sqrt{dNT})$.

3. Experiments: We conduct experiments on both synthetic and real world data, and see that compared to other regimes, our method outperforms well known algorithms in current literature.

The algorithm we propose does not use an optimism based principle on meta level. Instead, it utilizes a greedy approach. The interesting part is that the drawbacks of greedy algorithms, such as getting stuck in a suboptimal solution and over-exploit while under exploring, does not apply to the low rank setting we work on. The reason is that, the meta level exploration still exists even though the algorithm is greedy, through the noise on the episodic level and the way that meta level approximation works: re-estimating the parameters in $d-$dimensions.

1.3 Chapter 3: Bilinear Bandits with Arbitrary Arrivals

Consider an online recommendation problem, where users arrive in a platform, such as a streaming service or media or arbitrarily, and such platform wants to make a well-informed decision on which item to show to the incoming user based on the limited data access they have. This is a common recommendation phenomenon faced by many platforms, and hence the models and algorithms to solve this problem are very widely studied. They are solved mostly by using matrix completion methods, or certain bandit algorithms.

We are focusing on the linear bandit setting where the rewards are noisy with a mean that is linear in the unknown user vectors. More precisely, stochastic linear bandits is an online learning model that associates each action with a feature vector and assumes the mean reward is the inner product between the feature vector and an unknown parameter vector [1], [24], [25]. There are numerous applications that make use of this framework, such as serial webpage recommendations, pricing, or various other problems about modeling user behavior. Generally speaking, the idea is to learn these unknown parameters of users within a time frame called horizon.

Previous work on linear bandits has focused primarily on the case where this horizon is much larger than the dimension of the space. The main reason for this is that the minimax lower bounds are linear as opposed to sublinear in horizon in the worst case when the opposite is true, and the desired rate of learning may not happen in an adversarial setting. However, many real world applications deal with high dimensional settings in which the horizon may not be as large as these algorithms require. The framework we propose is what sheds some light into this problem: we can learn about users more efficiently across different games when the dimension is much larger and the underlying structure of these distinct parameters is low-rank. In other words, on a higher level we also learn about the lower dimensional space from which the parameters come from. We place this idea in the meta learning with bandits literature.

Meta bandits are a part of meta learning, where the agent has extra information on the structure behind the incoming parameters to the system and learns the parameters to that of the structure as well. In the bandit world, the algorithms also try to learn the underlying

information about these parameters so as to use it to discover the parameters themselves. We see different regimes of meta learning within the bandit literature such as models with a prior knowledge on the distribution over these parameters [21], or over the arms [49], meta learning over different bandit models [83], sparsity [39], low rank reward matrix where the parameters are tensors [57]. Our work fills the gap where the parameters come from an unknown lower dimensional subspace and the dimension we operate on can be much larger than the horizon. The bandit model described above has strong correspondence in real-world recommender system applications. Consider an online system where we see a series of users coming in, staying for a fixed amount of rounds, receiving recommendations, observing rewards and leaving. The platform can consider the rewards as implicit, such as the time spent, or explicit such as the binary 'click-no click' proxy measure. The problem the platform is concerned with is that, given that these users have unknown feature vectors coming from a lower dimensional subspace, what kind of approach can we take to learn more and/or better about the current and future users so as to maximize the cumulative reward. The problem reveals itself to be 2-fold: the individual parameter learning part, which is what most of bandit literature is focused on, and the meta learning of the subspace on a higher layer, which is learning the underlying structure these parameters come from.

There are certain caveats in solving a cold start problem by employing a bandit algorithm, which in fact is the common and the most advanced method, is that the arm (or item in this case) knowledge come from previously made observations, i.e. old data. Although matrix completion methods are quite pristine, once the latent space changes from dataset to dataset, the knowledge transfer does suffer from some loss. We are trying to solve a particular type of problem caused by this: arms have high dimensional estimated vectors, or the arms are new themselves which would result in a high dimensional setting, and that the true dimension is much lower than the ambient dimension.

The problem we are concerned with falls under a more realistic setting where $d \gg T^j$, where T^j is the amount of times a user j has appeared in the system. In a high dimensional setting, well known classical algorithms do not 'start working' until a certain amount of observations have been made, and play a randomly selected arm (or show a randomly selected item in this context), which hurt the overall performance of the recommender system.

To overcome this issue, we require a further common assumption: *the user vectors come from a lower dimensional subspace*. Notice that this assumption is not a newly-made one, since the entire matrix completion literature feeds from the assumption that the reward matrix has low rank structure. This, consequently, means the matrix whose rows are the user vectors in the system, has a low rank structure as well.

Contributions The contributions we make in this paper is 3-fold:

1. Model: We model this problem by using high dimensional meta linear bandits with arbitrary arrivals. The $d-$dimensional user vectors lay in a lower $s-$dimensional subspace. The rewards are bilinear in these unknown user vectors.

2. Algorithm: We propose a novel iterative algorithm based on a convex relaxation of a rank minimization problem. We then solve this problem by approximating it using SoftImpute, and further borrow from optimism based principles from matrix completion literature.

3. Experiments: We test our algorithm against well known methods from literature on a real-world data set from NetEase music streaming platform, containing over 50 million impressions. Our algorithm outperforms the benchmarks on various settings we test on. We further analyze the effect of the relationship between the ambient dimension and underlying rank.

Chapter 2

Disposable Linear Bandits

2.1 Introduction

Today's personalized online world comes with new types of problems of its own. Consider dynamic, user-centric systems such as dating websites, media platforms that provide movies, music and shows, online news articles or any kind of online advertisement. Such systems feed on the number of clicks or the amount of time a user spends on the platform, which necessitates a careful design of the underlying architecture that *recommends* products to users. Currently, the algorithmic systems behind such platforms work by learning the characteristics of items and users mathematically after a number of trials, and exploiting that information to further suggest products users might be interested in, consequently increasing the time users spend on the platform.

The key problem we address in this paper is for a particular subset of these platforms, in which each product has an individual budget bounding how many times that product can be recommended to one person. After the product *expires*, the system cannot recommend the product/person again. We refer to this property as *disposability* throughout the paper. The following two examples serve to illustrate this property in applications:

1. **Personalized Styling Services:** A new crop of online retailers has emerged in the last ten years offering extremely personalized assortments; examples including Stitch Fix, BeautyFIX, and Trunk Club. By and large, these companies work by offering, either

one-time or on a subscription basis, personalized "boxes" containing mostly clothing, jewelry, shoes, cosmetics, etc. These boxes are tailored to each user based on detailed information from an initial questionnaire, along with input from both human and algorithmic stylists. Upon receiving a box, users can choose to purchase any subset or the items, or even send the whole box back. These companies cannot re-send an item to a user, and so feedback on an item (whether or not it was purchased) is one-time. This item-selection process is a strict example of what we call disposability.

2. **Online Dating Platforms:** The success of any dating platform almost always depends on the quality of its recommendations, whether its revenue is tied to engagement or the number of successful "matches". Modern dating apps such as Tinder or Bumble, in an attempt to appeal to younger audiences, have introduced an additional dimension to the search process: users are shown other users' profiles in sequence. This means that recommendations are disposable with a single-use budget, as users prefer not to be shown the same person twice.

The current literature does not address the problem of disposable recommendations thoroughly. Instead, available methods rely on repetitive, trial-and-error recommendations, which as we have just described, often do not match real-world implementations of these systems. Therefore, the question we seek to answer is **how should recommendations be made under the disposability constraint?**

We will study this problem in the framework of bandit problems. In this class of problems, the primary trade-off is between so-called *exploration* and *exploitation*. For example on a news site, a manager reading business-related news might be suggested to read more articles about finance or politics, since other users with a similar personal history on this site have previously preferred such news over, say, celebrity or lifestyle articles. As the platform wants to maximize the news that this user consumes, it recommends items with a higher certainty of being clicked on. This is called *exploitation*: the process of exploiting known information and making the best move. However, users can be interested in a variety of topics or items, which the platform has no information beforehand. That brings us to *exploration* part of these algorithms. More often than not, the website gives users a suggestion of something

quite different from her past browsing pattern. This is only to explore what people might enjoy or not enjoy further, rather than the already available information.

In a standard bandit setting, algorithms usually recommend items multiple times in the process of estimating user preferences. These problems are very well-studied, and the literature now has close-to-optimal algorithms; we describe these further in Section 2.2. We give a generalization to the most common order-optimal algorithm that also extends well to the disposable setting.

Summary of Contributions: To summarize the points made earlier, this paper considers the problem of finding an effective algorithm to recommend items to users over time, with the goal of maximizing the cumulative reward, and with the restriction that an item may not be recommended more than a certain number of times, i.e. *budget*. We model this problem in the framework of stochastic linear multi-armed bandits, and after adding the restriction of disposability, call this problem *Disposable Linear Bandits*. Our primary observation is that this disposability constraint impacts both (a) the performance of algorithms known to be optimal when disposability is not imposed, and (b) the nature of algorithms that achieve meaningful theoretical (regret) guarantees. Our specific contributions are as follows:

1. We prove a minimax lower bound of $\Omega(\phi d\sqrt{T})$ on the regret of the disposable linear bandit problem which suggests that disposability should allow for lower regret than the standard linear bandit. In particular, while the dependence on the dimensionality d and time horizon T remains the same as in the standard linear bandit, a new term $\phi \in [0, 1]$ loosely captures the 'spread' of the optimal set of arms.

2. We propose an optimism-based algorithm, which generalizes the optimal upper confidence bound algorithm (called *LinUCB*) for the standard linear bandit. We prove an upper bound of $O(\sqrt{\bar{\gamma}}d\sqrt{T})$ on its regret, where γ is a parameter that again encodes a notion of the spread of the optimal arm set. Noting that our algorithm involves a computationally hard optimization formulation, we propose a fast alternating heuristic. While this heuristic is not guaranteed to find the true optimum, we observe that it performs well in practice when combined with a clever initialization based on similarity scores.

3. We evaluate our algorithm's performance on a recommendation task based on synthetically generated data. Compared to a number of benchmarks, including LinUCB and a natural modification of Thompson sampling, our algorithm (solved via the preceding heuristic) achieves as much as 10% lower regret against all competing algorithms.

The remainder of this paper is organized as follows. We conclude this section with a review of relevant literature. Section 2.2 introduces the problem formally and gives a hindsight of the existing LinUCB algorithm for linear bandits. In Section 3, we extend this LinUCB algorithm to an ideal case for disposable linear bandits, which is computationally hard to solve. That brings us to Section 4. In this section we provide a heuristic called *Alternating Heuristic*, which is the main contribution of this paper. In Section 5, we implement this heuristic along with LinUCB Algorithm, Thompson Sampling Adaptation and Greedy Algorithm on simulated data. Implementing the benchmarks and our method, we demonstrate that our method outperforms all other benchmarks. Section 6 concludes this paper with possible extensions for future work.

2.1.1 Literature Review

Our work is motivated, and draws upon, two large streams of literature: one on recommender systems and collaborative filtering, and one on bandit-type problems.

Recommender Systems: Especially after The Netflix Prize, a competition for finding the best collaborative filtering algorithm to predict user ratings, recommender systems in both academia and technology companies have gained immense popularity and been worked on extensively, almost to a perfection state.

There are mainly two types of filtering recommender systems use, content based filtering and collaborative filtering. Content based filtering recommends items based on a comparison between the content of the items and the user profile. This type of filtering is commonly used for spam detection, combined with various machine learning methods like Naive Bayes and Support Vector Machines such as in [36], [51] and [61].

Collaborative filtering, from which our work also feeds on as presented in Section 2.5, makes use of the idea that similar people will also prefer similar products. Similarity of users

is analyzed by Latent Semantic Analysis, details of which is explained thoroughly in [27]. Detecting possible changes in people's behaviors and adjusting the model accordingly were also studied by time-discounted models in [13] and [66]. Another standard method to use is Nearest Neighbor algorithm which takes the closest points to an item/user in space of features and makes inferences based on that information. [65] gives a brief overview of this methodology.

Companies, however, mostly use a hybrid approach that combine user-user and user-item information to get better performance. [60] shows that in some cases hybrid approach out-performs the case that one of the methods is used solely. In [34], they propose a scalable method switching between these different methods.

The problem of recommendation becomes more complicated when a new user is in the network. Even though the platform already has preferences of other users estimated, the strategy of recommending an item to this new user becomes an issue from an engineering perspective, which is called the *cold-start problem* in recommender systems literature. Cold-start problem is the problem in data-based automated systems, such as recommendation platforms, that the system does not have enough information/data about a user or an item, to make any inferences about it. Some folks focused especially on algorithms improving this *cold-start problem*. [14] , [74], [56] and many more that we do not include here, propose different heuristics and algorithms combined with collaborative filtering methods that improve this problem.

Bandits: Another method used to learn about preferences is Multi-Armed Bandits. The classical multi-armed bandit problem, the problem where a player has to make sequential decisions about which action to take given the past observations, was introduced by [64]. The problem is well studied for a couple of decades now, pioneered and put together by [35], who introduced the all-time famous *Gittins Index*. The known best algorithm for the classic simple MAB is the *Upper Confidence Bound (UCB)* algorithm by [48].

In the intersection of recommender systems and MAB literature, a natural approach is that the items to be recommended are modeled as arms of the bandits and each user is the player. How to use certain MAB techniques for recommendations are studied before, such as in [16], [80]. In recommendations setting, however, learning a user's preference is

most sought-after, so linear bandits, first introduced in [6], is an intuitively better method to capture the multi-dimensional relationship between the items and the users. The flavor of linear bandits is different from the classic MAB in the sense that rather than trying to estimate the arms with no prior information, we try to learn the player's characteristic vector by trying out new arms whose information is available. It is called linear MAB since the reward function is linear in player's feature vector. Linear multi-armed bandits have their own UCB algorithm with a similar logic to that of simple bandits, which is called LinUCB, LinReL, or OFUL. We call the algorithm LinUCB throughout this paper, and is explained more in detail in Section 2.3.

The studies we presented so far is merely the tip of the iceberg, and unlike this paper's focus, they are concerned with the bandits that are not restricted by disposable constraints. However, the disposable bandit idea, or similar ones, have been looked at in a few settings before. In [20], multi-armed bandits are used to model an assortment problem of fast-fashion retailers. In [30], the arms are irrevocable in the sense that at each time step the player can either keep playing the arm or decide to discard it. The player is allowed to play multiple arms up to a known cap. The authors solve the case by providing a heuristic, *packing heuristic*, which makes use of a ranking that depends on the remaining time and potential value. This case is different than ours, since our model has strict and deterministic budgets on arms and pulling multiple arms simultaneously is not feasible. In [23], the authors study the case that each arm has a *stochastic* remaining lifetime. They are inspired by online advertisements where ads have limited time to be shown. This is a different method since the availability time is not deterministic.

Introduced by [10] and then extended to the contextual version by [9], blocking bandits represent the case for which the arms become unavailable for a deterministic amount of time after they are played. Our case of disposable bandits with budgets of 1, however, cannot be seen as a special case of this model by simply making the unavailable time infinity. We emphasize two points here, one is in the case that a classic bandit is disposed after playing, since no information about that arm can be exploited later, every policy is the same with *random* policy. Therefore, this method is not a useful option. Second is that we do not face this issue since in the linear bandit problem, we have the information about arms upfront

and the parameter we estimate remains the same in between rounds.

A seemingly very similar approach that was studied in bandits is *budgeted bandits*. Budgeted bandits have been studied in both classic MAB and linear models in [76], Thompson Sampling on budgeted bandits in [75], cost-aware cascading bandits [31]. In all of these models, however, the budget is a common budget on all of the resources that arms might consume, whereas in our case, we consider independent budgets on arms. In [70, 26], the model is a classical MAB model except that each arm has an individual certain budget. Here we note that this actually is a type of disposable bandit where we have unit basis vectors as arms, the dimension is the total number of arms, and given that we clone each arm as many times as its given budget. Bandits-with-Knapsacks, as studied in [7, 3], are the types of bandits that consume a set of resources each time they are played.

Bandits that *deteriorate* or vanish by time have been studied in [30, 23, 54, 68]. In [30], the agent can either keep playing an arm or decide to discard it, which is a very different problem than ours, since we require discarding an arm immediately after the budget is exhausted. [23] has a stochastic time until when an arm is exhausted. [54] bandits have a decaying reward function depending on the number of times an arm is pulled. [45] has varying arm-sets over time, however, the structure differ significantly from ours, since we do not allow an unavailable action to be available again in the future, i.e. the available arm set is shrinking over time.

Finally, in operations literature, bandits with certain constraints have been used to model a variety of concepts and combinations of models: bandits with global constraints [4] where the model has concave objective functions and convex constraints, interactive marketing problems [11], restless bandits in which the state of arms change over time [12]. However, the methods and modeling technique used in these papers are fundamentally different from our model and hence, cannot be generalized or specialized into disposable linear bandits.

2.2 Model and Problem

The classical K-armed bandit problem is a famous reinforcement learning problem in which a player, at each discrete time step, pulls a single arm from the K possible choices, receives a

noisy reward which depends on that arm, and updates her information for future moves. The overall goal of the player is to maximize total reward earned at the end of the time horizon.

The type of bandit problems we focus on are called linear bandits. The setup of linear bandits is the same as above: at each time $t = 1, \ldots, T$, the player chooses an arm from a finite set of K arms $A \subset \mathbb{R}^d$ known by the player in advance, and receives a stochastic reward $r_t \in \mathbb{R}$. The reward function in the most simple k-armed bandit problems are Bernoulli random variables, whereas in *linear* bandits, the reward function is linear in some unknown $\theta^* \in \mathbb{R}^d$. More precisely, the reward observed after playing an arm a follows a linear function with mean-0 noise, i.e.

$$r_t = \langle a, \theta^* \rangle + \eta_t$$

where η_t is a σ^2 sub-Gaussian noise:

$$\mathbb{E}[\exp(\lambda \eta_t)] \leqslant \exp(\lambda^2 \sigma^2 / 2) \quad \forall \lambda \in \mathbb{R}.$$

In our framework, θ^* will represent the player's own feature vector, and is the object we try to estimate dynamically. We assume the following bounds on arms and θs: $\|a\| \leqslant L$ for all $a \in A$ and $\|\theta^*\| \leqslant L$.

We call the decision-making process of the player a *policy*. A policy π is any function that takes the previously observed outcomes, i.e. the arms that were played and the corresponding rewards, and maps it to an arm index to be played in the following period. More precisely, a policy at time t is a function of the form

$$\pi_t : \mathbb{R}^{t-1} \times A^{t-1} \mapsto \{1, \ldots, K\}$$

So the arm index that a policy chooses at each time step t is $\pi_t = \pi_t(r_1^\pi, \ldots, r_{t-1}^\pi, a_1, \ldots, a_{t-1})$, however, we will denote the arm chosen by this policy at time t as a_t^π for convenience.

The type of linear bandits we introduce is called *Disposable Linear Bandits*, or disposable bandits for short. Disposability is the case for which we have, for each arm i, an upper bound on the times we can play that arm, i.e. $\sum_{t=1}^{T} \mathbb{1}\{\pi_t = i\} \leqslant b_i$, for a given $b_i \in \mathbb{Z}^+$. Throughout the rest of this paper, to work on a more general case, we will take $b_i = 1$ for all arms w.l.o.g. The reasoning is that we can clone each arm a_i, b_i many times and we can make the new budget for each of those arms 1, i.e. we discard each arm after playing it. Then, the game

is naturally the same with the multi-budget case, only with more number of arms, some of which are identical to each other.

Due to disposability, now that we discard each arm after playing, we will also require that $K \geqslant T$, but it is usually safe to assume that $K \gg T$ regardless of the model we choose. As an example, on a dating platform, there may be thousands of people to be recommended to a user, but the user may only see a hundred recommendations over the entire time they use the platform.

Regret: The player's goal is to maximize the total *expected* reward earned over the T time steps, i.e. $\mathbb{E}\left[\sum_{t=1}^{T} r_t^{\pi}\right]$, but rather than measure this quantity directly, in bandit problems we usually instead optimize a quantity called *regret*. The regret of an algorithm is the difference between the total expected reward if the *best* actions were taken had θ^* been known, versus the total expected reward of the algorithm. Therefore the goal is to minimize cumulative regret, defined as:

$$R_T^{\pi} \equiv \mathbb{E}\left[\max_{|D|=T} \sum_{a \in D} \langle \theta^*, a \rangle - \sum_{t=1}^{T} r_t^{\pi}\right],$$

where D represents the decision set consisting of arms that can be played. The first term inside the expectation maximizes reward over all possible T-subsets of arms, and the second term is the reward of policy π.

Special Cases: Here we present two special cases: stochastic linear bandits and bandits with budgets as introduced in [70]. In disposable linear bandits, once we have $b_i = T$ for all $i = 1, \ldots, K$, the problem reduces to the classical linear bandit problem since one can play each arm T times in both cases, hence, disposability does not change the structure and the constraints of the game in this case.

k-armed bandits with budgets, on the other hand, are the regular $k-$armed bandits in which each arm has a deterministic budget b_i, capping the number of times that arm can be played. Although not obvious at first, the problem is a special case of disposable linear bandits when $\theta^* = [1]^K$ and the arms are the unit basis vectors $a_i = e_i$ with budget b_i. So each time the player plays an arm, he receives on expectation a reward of θ_j^*, if she played the arm e_j. Hence, the problem becomes estimating independent mean rewards of arms, which, clearly, is the classic multi armed bandit model.

2.2.1 Existing Approaches/Methods

Before we present the well-known algorithm for linear bandits in detail, we give how to construct a confidence region in $d-$dimension for an unknown $\theta \in \mathbb{R}^d$, given a history of noisy linear outcomes for this parameter.

Construction of Confidence Sets Let $\eta_1, \ldots, \eta_n$ be a sequence of independent 1-subgaussian random variables and $a_1^\pi, \ldots, a_t^\pi \in \mathbb{R}^d$ be a fixed sequence with $span(a_1^\pi, \ldots, a_t^\pi) = R^d$ and $r_1, \ldots, r_t$ be given by $r_i^\pi = \langle a_i^\pi, \theta^* \rangle + \eta_t$ for some $\theta^* \in \mathbb{R}^d$. Let $\hat{\theta}_{t+1}$ denote the least squares estimate of θ^*, based on the observed outcomes up until and including time t. Then,

$$\hat{\theta}_{t+1} = ((a_{1:t}^\pi)^T a_{1:t}^\pi + \lambda I)^{-1} (a_{1:t}^\pi)^\top r_{1:t}^\pi \tag{2.1}$$

where $\lambda > 0$ is a regularization parameter, $a_{1:t}^\pi$ is the matrix with rows $(a_1^\pi)^\top, (a_2^\pi)^\top, \ldots (a_t^\pi)^\top$ and similarly $r_{1:t}^\pi = (r_1^\pi, \ldots, r_t^\pi)^\top$.

Below we give the lemma that helps us construct a confidence set around this estimator $\hat{\theta}$. We first introduce the notation use. Let $V \in \mathbb{R}^{d \times d}$ and $u \in \mathbb{R}^d$, then, the $V-$ norm of a vector, $\| \cdot \|_V$, is defined as:

$$\|u\|_V := \sqrt{u^\top V u}$$

In our bandit setting, we will denote the matrix that keeps past information about the arms played up until time t as V_t:

$$V_t = V_0 + \sum_{i=1}^{t-1} a_i^\pi (a_i^\pi)^\top$$

where V_0 is a positive definite matrix, mostly $V_0 = \lambda I$ for some $\lambda > 0$. Hence, we can re-write the MLE estimator of θ^* as $\hat{\theta}_{t+1} = V_t^{-1} (a_{1:t}^\pi)^\top r_{1:t}^\pi$. We point that θ^* lies in a region concentrated around this $\hat{\theta}$, a confidence set, with some high probability. More precisely, [1] proved the following theorem on the construction of the confidence sets for θ^*.

Lemma 2.2.1 ([1], Theorem 2)**.** *(Confidence Ellipsoid) Let $V_0 = \lambda I, \lambda > 0$, and define $r_t = \langle a_t, \theta^* \rangle + \eta_t$ and assume that $\|\theta^*\| \leqslant S$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, for all $t \geqslant 0$, θ^* lies in the set*

$$\Theta_t = \left\{ \theta \in \mathbb{R}^d : \|\hat{\theta}_t - \theta\|_{V_{t-1}} \leqslant \sqrt{2 \log \left(\frac{\det(V_t)^{1/2} \det(\lambda I)^{-1/2}}{\delta} \right)} + \lambda^{1/2} S \right\}$$

Also, if for all $t \geqslant 1$, $\|a_t\| \leqslant L$, *then with probability at least* $1 - \delta$, *for all* $t \geqslant 0$, θ^* *lies in the set*

$$\Theta_t = \left\{ \theta \in \mathbb{R}^d : \|\hat{\theta}_t - \theta\|_{V_{t-1}} \leqslant \sqrt{d \log \left(\frac{1 + tL^2/\lambda}{\delta} \right)} + \lambda^{1/2} S \right\}$$

The confidence set defined above turns out to be an ellipsoid. We take the quantity that upper bounds $\|\hat{\theta}_t - \theta\|_{V_{t-1}}$ as a constant, β_t, and get

$$\Theta_t = \{ \theta \in \mathbb{R}^d : \|\hat{\theta}_t - \theta\|^2_{V_{t-1}} \leqslant \beta_t \} \tag{2.2}$$

We say that with probability $1 - \delta$, the true θ^* lies in this confidence ellipsoid Θ_t given above. Now, we make use of these sets to apply the optimism principle on linear bandits in the following.

LinUCB Algorithm The consensus best algorithm is called LinUCB [55], or similarly OFUL algorithm from [1]. OFUL stands for "optimism in the face of uncertainty in linear bandits," and the algorithm applies a principle that has proven to be effective in versions of bandit problems. This principle, or meta-strategy, works as follows

Among all available actions, choose the one with highest upside potential.

This principle should be contrasted with the greedy strategy, which at a high level chooses the action with highest *expected* potential. In the setting of linear bandits, the greedy algorithm calculates the least-squares estimator $\hat{\theta}$ calculated as in (2.1), and behaves as if $\theta^* = \hat{\theta}$. In contrast, LinUCB uses past observations to form a *confidence set/uncertainty set* Θ_t, and optimistically acts as if θ^* is equal to the best possible $\theta \in \Theta_t$ that will give the maximum possible reward with one of the arms.

The LinUCB algorithm balances exploration and exploitation in a natural way. It is certainly exploiting in the sense that the arm it plays is optimal with respect to some "reasonable" estimate of θ^*, i.e. one that lies in the confidence set. It is also exploring, as

it has the tendency to play arms in directions that have been previously observed the least. This is because the confidence set will be "wider" in those directions, and so the principle of optimism translates to favoring choices of θ lying in the wider parts, or equivalently, arms in those same directions.

So, at each time step t, LinUCB plays the following arm:

$$a_t^{LinUCB} = \underset{a \in A_t^{LinUCB}}{\operatorname{argmax}} \max_{\theta \in \Theta_{t-1}} \langle a, \theta \rangle$$

where A_t^{LinUCB} is the set of available arms at time t under policy LinUCB. We emphasize that the set of available arms in classic linear bandits remains the same through the game i.e. $A_1 = \cdots = A_T$, however, in disposable setting, A_t shrinks down each time since the arms are discarded.

The LinUCB algorithm chooses the arm that maximizes a linear function over an ellipsoid, of which we know the closed form [1]:

$$a_t^{LinUCB} = \underset{a \in A_t^{LinUCB}}{\operatorname{argmax}} \Big\{ \langle \hat{\theta}, a \rangle + \sqrt{\beta_t a^\top V_{t-1}^{-1} a} \Big\}.$$

In the formula of a_t^{LinUCB}, we can interpret the first term, $\langle \hat{\theta}_t, a \rangle$ as the exploiting greedy term, and the second one as the exploratory LinUCB term. In practice, β is the quantity we tune, and in our experiments also we take it as a constant tuning parameter throughout time, as also in [55].

Existing Regret Guarantees The classic linear bandit problem demonstrates a lower bound of $\Omega(d\sqrt{T})$ on the cumulative regret for *any* algorithm. The LinUCB algorithm has an upper bound of $\tilde{O}(d\sqrt{T})$, where $\tilde{O}$ hides the logarithmic factors, making the LinUCB algorithm order-optimal. The proof for these bounds can be found in Theorem 19.2 and Theorem 24.1 in [52].

As per the generalized algorithm we propose that can also deal with this disposability problem, although we still have $\sqrt{T}$ in the regret bounds, we manage to decrease the dimension d by a factor that depends on the reward distribution of the arm-set. We give the corresponding bounds in the following section.

2.3 Algorithm and Regret Guarantees

Now, we introduce the generalized version of LinUCB which takes into account the remaining moves (or budgets of arms) every period. The algorithm still feeds from the optimism principle, hence, it is an ideal generalization to the disposable linear bandit problem.

The algorithm works as follows. At each step, given the confidence set Θ_{t-1} for θ^*, instead of a single arm, the algorithm finds a *subset* of arms which will give the cumulative potentially highest reward at the end of remaining $T-t+1$ periods. It then plays an arm from that set uniformly at random and re-calculates the confidence set of θ.

Algorithm 1: Generalized LinUCB

Input: $\lambda > 0$

Initialization: $V_0 = \lambda I$, $\hat{\theta} = [0]^d$

Calculate Θ_0 according to (2.2).

for $t = 1, 2, \dots T$ **do**

$S_t^{\text{UCBG}} = \underset{\substack{|S|=T-t \\ S \subset A_t^{\text{UCBG}}}}{\operatorname{argmax}} \max_{\theta \in \Theta_{t-1}} \sum_{a \in S} a^\top \theta$

Play the arm a_t^{UCBG} chosen uniformly at random from S_t^{UCBG}

Observe reward: $r_t(a_t^{\text{UCBG}}) = \langle a_t^{\text{UCBG}}, \theta^* \rangle + \eta_t$.

Calculate $V_t = V_{t-1} + a_t^{\text{UCBG}} (a_t^{\text{UCBG}})^T$

Calculate Θ_t according to (2.2).

Remove arm a_t^{UCBG} from the future arm sets $A_{t+1}^{\text{UCBG}}, \dots, A_T^{\text{UCBG}}$.

end

There are two cases to analyze which back up the argument that this is a valid generalization of LinUCB. The first one is that this algorithm would behave exactly the same with LinUCB if it was implemented in a non-disposable setting. The only difference is that we do not remove the played arm from the possible action set. The set of *optimal* actions in non-disposable setting is playing the best arm, $A_1^\pi(1)$, T times in total. Reminding that linear bandits are the special case of disposable bandits in which there are T copies or each arm, we give the following:

$$S_t^{\text{UCBG}} = \underset{\substack{|S|=T-t+1 \\ S \subset A_t^{\text{UCBG}}}}{\operatorname{argmax}} \max_{\theta \in \Theta_{t-1}} \sum_{a \in S} a^\top \theta$$

Since we have *at least* $T-t+1$ copies of each arm at time t, the set S_t^{UCBG} consists of the copies of one arm that maximizes the potential reward over the confidence set. Choosing any arm from this set will result in the same arm, hence we can write:

$$a_t^{\text{UCBG}} = \underset{a \in A_t^{\text{UCBG}}}{\text{argmax}} \max_{\theta \in \Theta_{t-1}} \langle a, \theta \rangle = a_t^{LinUCB}$$

Indeed, A_t^{UCBG} is equivalent to A_t^{LinUCB}, since A_1^{UCBG}

Another point to be made is that in disposable setting, this algorithm behaves LinUCB-like in the last period, at time T, hence $|S_T^{UCBG}| = 1$. Therefore, the summation term will disappear as follows.

$$S_T^{UCBG} = \underset{\substack{|S|=1, \\ S \subset A_T}}{\text{argmax}} \max_{\theta \in \Theta_t} \sum_{a_t \in S} a_t^\top \theta = \underset{a_T \in A_T}{\text{argmax}} \max_{\theta \in \Theta} a_t^\top \theta = a_T^{LinUCB}$$

Finally, in the following subsections, we give the upper bound on UCBG, and a lower bound on disposable bandits.

2.3.1 Regret Analysis of Generalized UCB

Next we analyze the regret bound on this algorithm and show that as opposed to $O(d\sqrt{T})$ of standard linear bandits, UCBG enjoys a $O(d\sqrt{\bar{\gamma}\, T})$ upper bound, where $\bar{\gamma} \leqslant 1$ is a quantity capturing the reward difference among the best arm set through time. We say that *averaged arm* $\bar{A}$ is the vector averaged over all arms in the arm set A.

$$\bar{A} = \frac{1}{|A|} \sum_{a \in A} a \tag{2.3}$$

Using this notion, we introduce the following concept that appear in the upper bound.

Shrinkage Factor Let $A_t^\pi(k)$ denote the k^{th} best available arm at time t under policy π. Then, we define γ_t^π at any given time t as the ratio between the rewards of the best arm and the average over the best arm-set:

$$\gamma_t^\pi = \frac{\langle A_t^\pi(T-t+1), \theta^* \rangle}{\langle \frac{1}{T-t+1} \sum_{i=1}^{T-t+1} A_t^\pi(i), \theta^* \rangle} \tag{2.4}$$

Notice that $\gamma_t^\pi \leqslant 1$ for all t, and $\gamma_T^\pi = 1$ for any setting. We further define the concept of ρ_i as follows:

$$\|\overline{S}\| \leqslant \rho_i \text{ for all } S \subset A, |S| = T - i + 1 \tag{2.5}$$

We state that $\rho_T = 1$ for any policy, since all single arms have norm 1 in our model.

Theorem 2.3.1. *Suppose the following assumptions hold:*

1. *β_t, as defined in Theorem 2.2.1, is non-decreasing and $\beta_T \geqslant 1$.*

2. *Expected rewards are bounded, i.e. $0 \leqslant r_t \leqslant 1$.*

3. *For each $t = 1, \ldots, T$, with probability $1 - \delta$, $\theta^* \in \Theta_t$, where Θ_t is as in (2.2).*

Then for arms $a \in \mathbb{R}^d$ for all $a \in A_t$, and $t = 1, \ldots, T$, with probability $1 - \delta$, the regret of generalized UCB satisfies

$$R_T^{\text{UCBG}} = O\left(d\sqrt{\overline{\gamma\rho}\, T}\right)$$

where $\overline{\gamma\rho} = \frac{1}{T}\sum_{t=1}^{T} \gamma_t\rho_t$ and γ_t at time t is as defined in Equation (2.4) and similarly, ρ_t is as defined in Equation (2.5) .

Proof. The crux of the proof depends on the idea that we *do not* punish playing those arms that the optimal algorithm would play at some time. So we give the following lemma, which provides another way of bookkeeping for regret: The instantaneous pseudo regret is the positive part of the difference between the $T - t + 1^{st}$ best arm among the available arms at time t and the arm algorithm plays.

Lemma 2.3.2. *In the disposable bandit setting, the traditional definition of regret can be re-written as*

$$\sum_{i=1}^{T}\langle A_1(i), \theta^*\rangle - \sum_{t=1}^{T}\langle a_t, \theta^*\rangle = \sum_{t=1}^{T}\left(\langle A_t^\pi(T-t+1), \theta^*\rangle - \langle a_t^\pi, \theta^*\rangle\right)^+$$

Proof. Proof of Lemma 2.3.2

We need to show that the right hand side quantity, $\sum_{t=1}^{T}\left(\langle A_t^\pi(T-t+1), \theta^*\rangle - \langle a_t^\pi, \theta^*\rangle\right)^+$ is equal to the total regret, where $A_t^\pi(T-t+1)$ denotes the $(T-t+1)^{st}$ best arm among the re-

maining arms at time t under policy π. Let us call the quantity $(\langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle)^+$ at time t, *pseudo regret*, denoted by Q_t.

We analyze three different cases:

Case 1: Suppose the policy π does not play *any* of the optimal arms. In this case, the expected reward of a played arm will always be less than that of the $A_t^\pi(T-t+1)$. Hence, the immediate pseudo regret, Q_t is the same as the traditional immediate regret, i.e. $Q_t = \langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle$. So we omit the plus sign here. This yields the equality:

$$\begin{aligned}\sum_{t=1}^{T} (\langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle)^+ &= \sum_{t=1}^{T} \langle A_t^\pi(T-t+1), \theta^* \rangle - \sum_{t=1}^{T} \langle a_t^\pi, \theta^* \rangle \\ &= \sum_{t=1}^{T} \langle A_1^\pi(t), \theta^* \rangle - \sum_{t=1}^{T} \langle a_t^\pi, \theta^* \rangle\end{aligned}$$

where the last equality follows from the fact that the set of T best arms does not change throughout time, since the policy π always played a worse-off arm.

Case 2: The algorithm always plays one of the optimal arms, though not necessarily in a particular order. Then, $(\langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle)^+ = 0$ for all $t = 1, \ldots, T$ since $A_t^\pi(T-t+1)$ will always have less expected reward than the arm algorithm has chosen. Since there is also zero traditional cumulative regret, the two quantities are equivalent.

Case 3: The algorithm plays *some* of the best arms. The quantity $(\langle A_t^\pi(T-t+1) - a_t^\pi, \theta^* \rangle)^+$ will be 0 for those instances the policy π plays a better arm than $A_t^\pi(T-t+1)$, i.e.

$$\langle a_t^\pi, \theta^* \rangle > \langle A_t^\pi(T-t+1), \theta^* \rangle.$$

For all other instances, $(\langle A_t^\pi(T-t+1) - a_t^\pi, \theta^* \rangle)^+ \geqslant 0$, hence we can omit the plus sign for these. Ultimately, we make the following replacement in the formula for the times an algorithm plays a strictly better arm, so that we can omit the plus sign completely.

$$(\langle A_t^\pi(T-t+1) - a_t^\pi, \theta^* \rangle)^+ = 0 = \langle a_t^\pi - a_t^\pi, \theta^* \rangle$$

Let T_1 be the set of instances that the policy π plays a strictly better arm than $A_t^\pi(T-t+1)$,

and $T_2 = [T] \backslash T_1$.

$$\begin{aligned}
\sum_{t=1}^{T} (\langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t, \theta^* \rangle)^+ &= \sum_{t \in T_1} (\langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle)^+ \\
&+ \sum_{t \in T_2} (\langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle) \\
&= \sum_{t \in T_1} (\langle a_t^\pi, \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle) + \sum_{t \in T_2} \langle A_t^\pi(T-t+1), \theta^* \rangle - \langle a_t^\pi, \theta^* \rangle \\
&= \sum_{t \in T_1} \langle a_t^\pi, \theta^* \rangle + \sum_{t \in T_2} \langle A_t^\pi(T-t+1), \theta^* \rangle - \sum_{t=1}^{T} \langle a_t^\pi, \theta^* \rangle
\end{aligned}$$

Now all we need to show is that the sets $\{a_t^\pi | t \in T_1\}$ and $\{A_t^\pi(T-t+1) | t \in T_2\}$ do not have any elements in common. Notice that both sets are subsets of $\{A_1(t) | t \in [T]\}$. Let b_t, under policy π, be the rank of arm $A_t^\pi(T-t+1)$, i.e. $A_1^\pi(b_t) = A_t^\pi(T-t+1)$. Then, naturally $b_1 = T$. For an intermediate b_{t+1}, there are two cases to consider: when the policy π plays a strictly better arm than $A_1^\pi(b_t)$ and when it plays a worse arm. Let $\tilde{t}$ denote the first time π plays a strictly better arm. At time $\tilde{t}+1$, $A_{\tilde{t}+1}^\pi(T-\tilde{t}+1) = A_{\tilde{t}+1}^\pi(T-\tilde{t})$ *again* since among the remaining arms, the $(T-\tilde{t})^{th}$ arm is the $(T-\tilde{t}+1)^{st}$ arm at time $\tilde{t}$ due to π playing and disposing a good arm. Then $b_{\tilde{t}+1} = b_{\tilde{t}}$. So in the case that an algorithm plays a better arm, say k times consecutively, then b_t will be the same for the next k times also. If the policy π plays a worse arm at time $\tilde{t}$, then at time $\tilde{t}+1$, $b_{\tilde{t}+1}$ will be the next available arm with the largest rank strictly smaller than $b_{\tilde{t}}$. We define set Π_t as the set of the ranks of those arms policy π has played until and including time t.

$$\Pi_t = \{\pi_1, \ldots, \pi_t\}$$

Then we write the following recursive relationship:

$$b_{t+1} = \begin{cases} b_t & \text{if } t \in T_1 \\ \max\{\{1, \ldots, b_t - 1\} \backslash \Pi_t\} & \text{if } t \in T_2 \end{cases}$$

In summary, the sequence b_t repeats itself if policy π plays one of the good arms at time t, causing $b_{t+1} = b_t$. Otherwise, it picks the next one among the available arms.

Now what is left is to show that the following rank sequence for $t = 1, \ldots, T$ is indeed $\{1, \ldots, T\}$ up to permutation.

$$c_t = \begin{cases} \pi_t & \text{if } t \in T_1 \\ b_t & \text{if } t \in T_2 \end{cases} \tag{2.6}$$

Naturally $c_t \leqslant T$ for all t, since for $t \in T_1$, we have $c_t = \pi_t < b_t \leqslant T$ and for $t \in T_2$, we have $c_t = b_t \leqslant T$. Now, finally, we show that c_t never repeats itself, i.e. for a given time i, $c_i \neq c_j$ for $j < i$ w.l.o.g.

- Case $c_i = \pi_i$: In other words, $\pi_i < b_i$. If a previous c_j is π_j, then due to disposability, $\pi_i \neq \pi_j$, hence, $c_i \neq c_j$. If a previous $c_j = b_j$, then, we use the non-increasing property of the sequence b_t and write $b_j \geqslant b_i > \pi_i$, leading to $\pi_i \neq b_j$ and therefore $c_i \neq c_j$.

- Case $c_i = b_i$: In other words, $\pi_i \geqslant b_i$. At time j, c_j is either b_j or π_j. If $j \in T_1$, we have $c_j = \pi_j$ and $\pi_j < b_j$. It is easy to see that b_i can never be equal to π_j since b_t is defined on available arms and $i > j$, so π_j is not among the available arms at time i. If $j \in T_2$ then we have $c_j = b_j$ and $\pi_j \geqslant b_j \geqslant b_i$. From the definition of b, we know that for $j \in T_2$, $b_{j+1} \leqslant b_j - 1$ and that $i \geqslant j + 1$, hence $b_i \leqslant b_{j+1} < b_j$, showing $b_i \neq b_j$.

We have shown that c_t as defined in (2.6) has T unique ranks. Now what is left is to re-write the final expression we have for the new book-keeping method:

$$\begin{aligned}
\sum_{t\in T_1}\langle a_t^\pi, \theta^*\rangle + \sum_{t\in T_2}\langle A_t^\pi(T-t+1), \theta^*\rangle - \sum_{t=1}^{T}\langle a_t^\pi, \theta^*\rangle &= \sum_{t\in T_1}\langle A_1^\pi(c_t), \theta^*\rangle + \sum_{t\in T_2}\langle A_1^\pi(c_t), \theta^*\rangle - \sum_{t=1}^{T}\langle a_t^\pi, \theta^*\rangle \\
&= \sum_{t=1}^{T}\langle A_1^\pi(c_t), \theta^*\rangle - \sum_{t=1}^{T}\langle a_t^\pi, \theta^*\rangle \\
&= \sum_{t=1}^{T}\langle A_1^\pi(t), \theta^*\rangle - \sum_{t=1}^{T}\langle a_t^\pi, \theta^*\rangle
\end{aligned}$$

where the last equality follows from the fact that re-ordering of the arm sequence of a policy will not change the cumulative expected reward.

■

s ■

Brief Analysis of Upper Bound: UCBG upper bound behaves similarly with LinUCB, except to a fraction of $\sqrt{\overline{\gamma\rho}}$, a quantity that depends on the reward difference among the arm set. Relating back to linear bandits vs disposable linear bandits in which we have T copy of each arm, one can see that $\overline{\gamma\rho} = 1$ in the special case of linear bandits, matching the known upper bound $\tilde{O}(d\sqrt{T})$ and indeed, reassuring that this is a generalization of UCB algorithm. Intuitively we can say that since we do not punish the times that the algorithm plays an arm among $A_1(1:T)$, which has a set reward difference of the factor $\gamma_t\rho_t$, the regret we observe also *shrinks* accordingly.

Corollary 2.3.3. *For a set of distinct arms for which the following holds for all i, j:*

$$\langle a_i, a_j \rangle \leqslant \tilde{\gamma}$$

we have a looser general upper bound of $O(d\sqrt{T\tilde{\gamma}})$. One can interpret $\tilde{\gamma}$ as the quantity that captures how close at maximum two arms are in the arm set we are given.

The corollary follows from the fact that $\tilde{\gamma}$ is an upper bound on any reward ratio between any 2 arms: hence $\tilde{\gamma} \geqslant \gamma_i\rho_i$ for all $i = 1, \ldots, T$, and leading to $\tilde{\gamma} \geqslant \overline{\gamma\rho}$.

2.3.2 Lower Bound

In this subsection we analyze an instance for which we prove a lower bound of $\Omega(\phi d\sqrt{T})$, where ϕ is defined to be the ratio between the lowest and highest mean rewards among the set of T optimal arms. More specifically, taking $A_1(i)$ as the i^{th} reward-wise best arm at the beginning of the game, we define ϕ as:

$$\phi = \frac{\langle A_1(T), \theta^* \rangle}{\langle A_1(1), \theta^* \rangle} \tag{2.7}$$

Proposition 2.3.4. *Assume that $2^d \geqslant T$, and let the arm set be $\mathcal{A} = \{-1, 1\}^d$. For any algorithm, there exists $\theta \in \Theta = \{-\sqrt{1/T}, \sqrt{1/T}\}^d$ such that we have the following lower*

bound on regret:

$$\sum_{t=1}^{T} R_t(\mathcal{A}, \theta) \geqslant C\phi d\sqrt{T},$$

where C is a universal constant.

Proof. The idea behind this new lower bound, however, is that we compare the mistakes that have to be done by the ebst algorithm and any other algorithm. Since the best algorithm has a set of good actions it takes, the rewards of which changes according to the set, the final regret we can get also decreases by a factor that depends on the range of rewards in the optimal decision set.

The proof is similar to that of the lower bound for non-disposable linear bandits (e.g. Theorem 24.1 in [52]). Recall the setup: we assume that $2^d \geqslant T$, and let the arm set be $\mathcal{A} = \{-1, 1\}^d$. Finally, define the set $\Theta = \{-\sqrt{1/T}, \sqrt{1/T}\}^d$, from which θ will be chosen. The noise will be assumed to come from the standard Gaussian distribution.

For any $\theta \in \Theta$, let k be defined as

$$k = \max_t \sum_{j=1}^{d} \mathbb{1}\{\text{sign}(a_{\pi_t^*,j}) \neq \text{sign}(\theta_j)\}$$

where the notation $a_{i,j}$ denotes the j-th coordinate of a_i. In words, k is the number of dimensions along which the worst arm among (any choice of) A^* differs from θ in sign. Note that, by symmetry, k is equal for all $\theta \in \Theta$, so k is only a function of d and T. Since there exists an arm which matches θ in all dimensions in sign, ϕ (as defined in (2.7)) is equal to $\phi = \frac{d-k}{d}$.

Now fix any policy π and $\theta \in \Theta$, and define $p_{\theta,i}$ to be the likelihood that at least $n/2$ arms selected by this policy differ in sign from θ along dimension i, i.e.

$$p_{\theta,i} = \mathbb{P}_\theta \left(\sum_{t=1}^{T} \mathbb{1}\{\text{sign}(a_{\pi_t,i}) \neq \text{sign}(\theta_i)\} \geqslant n/2 \right).$$

Now fix a dimension i. For any θ and θ' such that $\theta' = \theta$ except $\theta'_i = -\theta_i$, Pinsker's

inequality (stated as Lemma 2.3.5 below) implies that:

$$p_{\theta,i} + p_{\theta',i} \geqslant \frac{1}{2}\exp\left(-\frac{1}{2}\sum_{t=1}^{T}\langle a_{\pi_t}, \theta - \theta'\rangle^2\right) = \frac{1}{2}e^{-2} \tag{2.8}$$

Let the notation $\sum_{\theta_{-i}}$ to denote summation over $\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_d \in \left\{\pm\sqrt{1/T}\right\}^{d-1}$. We obtain the following lower bound for $\sum_{i=1}^{d} p_{\theta,i}$:

$$\begin{aligned}
\sum_{\theta\in\Theta} 2^{-d}\sum_{i=1}^{d} p_{\theta,i} &= \sum_{i=1}^{d}\sum_{\theta_{-i}} 2^{-d} \sum_{\theta_i\in\left\{\pm\sqrt{1/n}\right\}} p_{\theta,i} \\
&\geqslant \sum_{i=1}^{d}\sum_{\theta_{-i}} 2^{-d}\frac{1}{2}\exp\left(-2\right) \\
&= \frac{d}{4}\exp\left(-2\right).
\end{aligned}$$

where the inequality follows from (2.8). This implies that there exists at least one θ for which

$$\sum_{i=1}^{d} p_{\theta,i} \geqslant \frac{d}{4}\exp\left(-2\right). \tag{2.9}$$

For the remainder of the proof, fix θ^* to be one such θ.

Now fix any optimal sequence of arms $a_{\pi_t^*}$ such that $a_{\pi_1^*,i} = \theta_i^*$ for all $i = 1, \ldots, d$, i.e. the first arm in this sequence is the best. The total regret can be decomposed as follows,

$$\sum_{t=1}^{T} R_t = \sum_{t=1}^{T}\mathbb{E}_\theta\left[\langle a_{\pi_t^*} - a_{\pi_t}, \theta^*\rangle\right] = \sum_{t=1}^{T}\mathbb{E}[\langle a_{\pi_1^*} - a_{\pi_t}, \theta^*\rangle] - \sum_{t=1}^{T}\mathbb{E}[\langle a_{\pi_1^*} - a_{\pi_t^*}, \theta^*\rangle],$$

which leads to:

$$\begin{aligned}
\sum_{t=1}^{T} R_t &= 2\sqrt{\frac{1}{T}}\sum_{t=1}^{T}\sum_{i=1}^{d}\mathbb{P}\left(\text{sign}(a_{\pi_t,i}) \neq \text{sign}(\theta_i^*)\right) - 2\sqrt{\frac{1}{T}}\sum_{t=1}^{T}\sum_{i=1}^{d}\mathbb{1}\left\{sign(a_{\pi_t^*,i}) \neq sign(\theta_i^*)\right\} \\
&\geqslant \sqrt{T}\sum_{i=1}^{d}\mathbb{P}\left(\sum_{t=1}^{T}\mathbb{1}\{\text{sign}(a_{\pi_t,i}) \neq \text{sign}(\theta_i^*)\} \geqslant T/2\right) - 2k\sqrt{T} \\
&= \sqrt{T}\sum_{i=1}^{d} p_{\theta,i} - 2k\sqrt{T} \\
&= \Omega(d\phi\sqrt{T}).
\end{aligned}$$

The first line is due to our fixing $a_{\pi_1^*}$ to be the best arm, and the second line comes from applying the definition of k as the maximal number of dimensions along which any $a_{\pi_t^*}$ disagrees with θ in sign. The fourth line comes from (2.9).

Lemma 2.3.5 ([72], Lemma 2.6)**.** *Let P and Q be probability measures on the same measurable space $(\Omega, \mathcal{F})$ and let $A \in \mathcal{F}$ be an arbitrary event. Then,*

$$P(A) + Q(A^c) \geqslant \frac{1}{2}\exp(-\text{KL}(P, Q)).$$

■

Brief Analysis of Lower Bound: The lower bound behaves similarly with that of LinUCB. Connecting back to the original problem of linear bandits, the special case when we have T-many of each arm, we see that $\phi + t = 1$ for all t, or equivalently, $k = 0$, i.e. the reward of $A_1(1)$ and $A_1(T)$ are the same, which in turn gives us the same lower bound $\Omega(d\sqrt{T})$. Also, notice that if $2^d = T$, i.e. we play all arms, $k = d$, and lower bound becomes 0, as expected.

2.4 Heuristics

The ideal algorithm UCBG given in Algorithm 1 tries to find the set S that maximizes the function $f(\theta, S) = \sum_{a \in S} a^\top \theta$ over the confidence ball. In other words, it solves the following

optimization problem each time:

$$\max_{\theta \in \Theta_{t-1}} \max_{|S|=T-t+1} \sum_{a \in S} a^\top \theta,$$

However, this amounts to calculating exponentially many subsets, since the algorithm would have to go over all $\binom{K}{T-t+1}$ subsets, and calculate the UCBG value of them, making the application of this algorithm computationally intractable. Hereby, we present a heuristic approach that was inspired by the UCBG value of an arm, based on the idea that there exists an arm close to a given θ. This assumption is not unrealistic, since in recommendations, we usually work with masses of items and therefore, the product space is quite dense.

We start by defining a function of θ, measuring its possible highest reward. In other words, let $g(\theta) = \max_{|S|=T-t+1} \sum_{a \in S} a^\top \theta$. Then g is a function that gives the maximum cumulative inner product of this θ and a subset of $T-t+1$ arms among the current available arm-set. Further, for any given θ, let $a_i(\theta)$ be the i^{th} closest arm to θ in Euclidean distance. Reminding that the arms and θ are on the unit ball, we make the following reduction:

$$g(\theta) = \max_{|S|=T-t+1} \sum_{a \in S} a^\top \theta = \sum_{i=1}^{T-t+1} a_i(\theta)^\top \theta$$

This equation mainly indicates that the subset we seek, which will maximize the cumulative inner product, is the subset of arms that are closest to θ in Euclidean distance. We re-write this expression by singling out the closest arm, $a_1(\theta)$.

$$g(\theta) = (a_1(\theta))^\top \theta + \sum_{i=2}^{T-t+1} (a_i(\theta))^\top \theta$$

Now, the main trick of this calculation is to make the substitution $a_1(\theta) \approx \theta$. We remind that the norm of θ is 1, and make this substitution twice in the following statement.

$$g(\theta) \approx (a_1(\theta))^\top \theta + \sum_{i=2}^{T-t+1} (a_i(a_1(\theta)))^\top \theta \approx (a_1(\theta))^\top \theta + \sum_{i=2}^{T-t+1} (a_i(a_1(\theta)))^\top a_1(\theta)$$

We emphasize that $a_1(a_1(\theta)) = a_1(\theta)$, hence the second term is actually the summation

of inner products of the closest arm to θ with the $T-t$ closest arms to itself. The final expression we have mainly says that in a dense arm-set, the maximum cumulative reward of a set of arms for a given θ is close to that of the single best arm and the summation of the inner products with the arms around it. We call the second term the *similarity score* of an arm a, denoted $S(a)$, and get

$$(a_1(\theta))^\top \theta + \sum_{i=2}^{T-t+1} (a_i(a_1(\theta)))^\top a_1(\theta) = a_1(\theta)^\top \theta + S(a_1(\theta))$$

Re-stating that we have unit vectors, this is indeed the cumulative cosine similarity distance which measures how different two arms are direction-wise. Hence the closer the arms are, the greater the summation becomes. Now, instead of maximizing this final quantity over the ellipsoid, we iterate and take the maximum over all arms. Intuitively what this means is that the set of arms to be chosen should be similar to each other (i.e. cosine similarity) so that the algorithm explores those arms that it can later exploit, which is crucial due to the disposable nature of the problem. In other words, we give less weight to explore those with little similarity to other arms around them, and decide to learn less in those directions.

Before we explain the method in detail, we give the two functions Similarity(v, A, s) and Closest(v, A, s) more formally. Similarity(v, A, s) gives the cumulative similarity score of arm v with the set of arms that are closest to it in cosine-similarity measure. Closest(v, s, A), on the other hand, gives a set of arms with size s from A with the highest inner products with vector v.

$$\text{Similarity}(v, A, s) = \max_{S \subset A, |S|=s} \sum_{a_s \in S} \langle v, a_s \rangle$$

$$\text{Closest}(v, A, s) = \operatorname*{argmax}_{S \subset A, |S|=s} \sum_{a_s \in S} \langle v, a_s \rangle$$

The procedure of Alternating Heuristic is as follows. We first choose an arm a_t with the highest UCB + Similarity index. Intuitively this means find an arm who has a high reward estimate *and* a high number of arms close to it. Then we find the set of arms closest to this arm, S_t^{AH}. Notice that the θ that will maximize the optimistic reward for this set is different from that of a_t. So we find the $\tilde{\theta} \in \Theta_{t-1}$ that maximizes $(\overline{S_t^{AH}})^T \tilde{\theta}$. We use the fact that

maximizing UCB($\overline{S_t^{AH}}$) is equivalent to maximizing UCBG(S_t^{AH}) = $\max_{\theta \in \Theta_{t-1}} \sum_{a_t \in S_t^{AH}} a_t^\top \theta$. For a fixed θ, we can find the arm set to maximize $g(\theta)$ by choosing the arms closest to it, which gives us the set S_t^θ. We iterate until the two arm sets, S_t^θ and S_t^{AH} are equivalent. Finally, choose the arm in S_t^{AH} with the maximum UCB index to play. Note that in each iteration we increase the objective function value f we try to maximize.We point out that f is a non-concave function, so the heuristic will converge to a set of arms that is locally optimal, not globally. It is crucial to state that the heuristic always converges for two reasons: we have a finite set of arms and there always exits an instance for which f is maximized.

Algorithm 2: Alternating Heuristic

Input: $\lambda, \alpha, c > 0$
Initialization: $V = \lambda I, \hat{\theta}_0 = [0]^d$
for $t := 1, 2, \ldots T$ **do**
  $a_t = \underset{a \in A_t^{AH}}{\operatorname{argmax}} \langle \hat{\theta}_{t-1}, a \rangle + c\sqrt{a^T V_{t-1}^{-1} a} + \alpha \, \text{Similarity}(a, A_t^{AH}, T-t)$
  $S_t^{AH} = \text{Closest}(a_t, A_t^{AH}, T-t)$
  $\tilde{\theta} = \theta^{\text{UCB}}(\overline{S_t^{AH}})$
  $S_t^\theta = \text{Closest}(\tilde{\theta}, A_t^{AH}, T-t)$
  while $S_t^{AH} \neq S_t^\theta$ **do**
    $S_t^{AH} = \text{Closest}(\overline{S_t^\theta}, A_t^{AH}, T-t)$
    $\tilde{\theta} = \theta^{\text{UCB}}(\overline{S_t^{AH}})$
    $S_t^\theta = \text{Closest}(\tilde{\theta}, A_t^{AH}, T-t)$
  end
  Play the arm $a_t^{AH} = \underset{a \in S_t^\theta}{\operatorname{argmax}} \langle \hat{\theta}_{t-1}, a \rangle + c\sqrt{a^T V_{t-1}^{-1} a}$
  Observe reward: $r_t^{AH} = \langle a_t^{AH}, \theta^* \rangle + \eta_t$
  $V_t = V_{t-1} + a_t^{AH}(a_t^{AH})^T$
  $X = [X, a_t^{AH}]$
  $Y = [Y, r_t^{AH}]$
  Calculate $\hat{\theta}_t = V_t^{-1} X^T Y$
  Remove arm a_t^{AH} from the future arm sets $A_{t+1}^{AH}, \ldots, A_T^{AH}$.
end

As stated, the alternating procedure we apply to a given subset always increases the function f we try to maximize. However, one caveat could come from the method we use to initialize the arm set. To address this issue and see what is the best way to initialize the set, we test three different initialization to compare the objective function and the convergence rate. Although the data and experiments are explained in detail in Section 2.5, below we

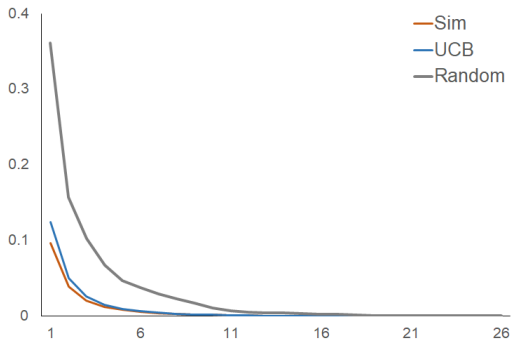
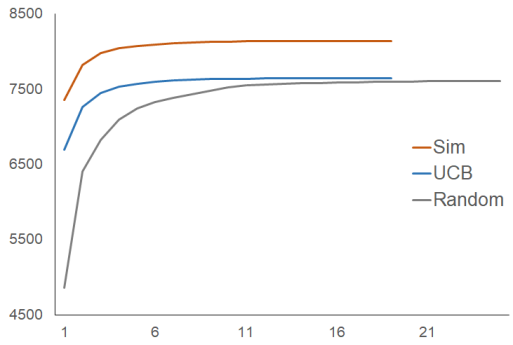


Figure 2-1: Performance and convergence rate of alternating heuristic with three different initializations. Results are averaged over 200 instances, all at period $t = 5$. *Performance (Left):* Objective function vs. iteration number. *Convergence Rate (Right):* Gap fraction to eventual converged value vs. iteration number.

provide the comparisons of the objective function between those different initial arm sets given to the Alternating Heuristic. These trends were observed during $t = 5$, for other time-step trends refer to the Appendix.



We test 3 different initializations over the arm set. *Random*: Choose $T - t + 1$ arms randomly and iterate until convergence. As seen naturally from the figures, this takes more iterations and converges to a lower function value. *UCB*: Choose $T - t + 1$ arms by $\underset{S \in A_t, |S| = T-t+1}{\operatorname{argmax}} \sum_{a \in S} \text{UCB}(a)$, i.e. arms with the highest combined UCB values. This method is significantly better than random initialization, but still has a lower final value than that of Similarity initialization. *Similarity*: Choose $T - t + 1$ arms by maximizing the similarity score of one arm, and choosing $T - t$ arms closest to it. This initialization not only have the highest function value it converges, but also converges faster than random, similar to that of LinUCB.

2.5 Experiments

In this section, we describe the procedure we followed to test our heuristic and algorithms on a real-world data from a popular online dating platform. We cannot disclose the company name here explicitly for privacy reasons, however, we will explain how the app works and describe the data in detail. The app works as follows. A user is shown two cards from the opposite gender each day and he/she can decide to choose one and send a *proposal*. With each proposal made, the platform earns a certain amount of money. In the next step, those who get proposals can either accept or reject the proposal, which initiates the interaction process. The goal of the platform is, clearly, match as many users as possible.

The data from this platform contains 199K male users and 44K female users. It has the information of users themselves such as age, gender, location, occupation, physical appearance, popularity among other users etc. It also has 2-year worth of transactions between users such as number of messages, the duration of contact, who sent the proposal or who accepted a proposal etc. For our purposes, we are only interested in the second part since side information is not included in our scope.

We extract from this data a main matrix of men vs women, where men are the rows of this matrix $i = 1, \dots, K$, and women are the columns $j = 1, \dots, K$. We chose same number of men and women for simplicity purposes. Each entry of this matrix represents the probability of two people matching. So man i and woman j matches with probability M_{ij}, where $M \in \mathbb{R}^{K \times K}$ is the main matrix we work on, which denotes the *reward matrix*. We take the first 5000 most active users for our experimental purposes, hence $K = 5000$ for the rest of this section.

Naturally, the matrix is not even close to complete. To solve this issue, we employ Singular Value Decomposition method to complete it. Consider each entry as a reward that is linear in men and women's unknown vectors. So we decompose the matrix M as

$$M = \Theta^\top A$$

where $A, \Theta \in \mathbb{R}^{K \times d}$ and $\Theta_j \in \mathbb{R}^d$ represents the vector of each man, similarly A_j of each woman. In our data, the dimension is 16, $d = 16$. Notice that d is the rank of the matrix and it is way smaller than the size, $d \ll K$. By using this, we approximate all A_i and Θ_j

such that $M_{ij} = \Theta_i^\top A_j$. We treat these entries as the true mean reward we would get from recommending person j to person i, i.e. each Θ_i is a different θ. However, the entries observe have mean-zero noise η_{ij}:

$$R_{ij} = \langle \Theta_i, A_j \rangle + \eta_{ij}$$

In the following we give the adaptations for disposable setting for the benchmarks we compare Alternating Heuristic to: Greedy Algorithm, LinUCB and Thompson Sampling.

2.5.1 Disposable Setting Adaptations

For experimental purposes, we modified the current well-known algorithms by adding the disposability constraint. The modification is straightforward: after playing an arm, we remove it from the future arm sets. We remind that we still work with the more general case in which the arms have a budget of 1, hence, discarded after a single pull.

Greedy Algorithm Greedy algorithm takes the maximum likelihood estimator of θ and behaves as if it is the true θ. In Gaussian setting, we give the MLE estimate of θ as follows:

$$\hat{\theta}_t = V_t^{-1} X^T Y$$

where X is the matrix with rows $(a_1^{Greedy})^T, \ldots, (a_{t-1}^{Greedy})^T$, and Y is the vector of rewards observed so far, i.e. $r_1^{Greedy}, \ldots, r_{t-1}^{Greedy}$. Greedy algorithm chooses the arm that will maximize the inner product with this estimator of θ^*, $\hat{\theta}$. In disposable setting experiments, we remove the arm Greedy strategy has played from the available arm set for the rest of the game.

LinUCB Algorithm We use the well-known LinUCB algorithm with a modification that we remove the arm we play at a time period from the future available arm sets. We tune the c value for the following LinUCB index for a given arm:

$$LinUCB(a) = a^T \hat{\theta} + c\sqrt{a^T V^{-1} a}$$

In the results reported, that value of c that gives the smallest regret was chosen for LinUCB.

Thompson Sampling The version of Thompson Sampling we work with use multivariate Gaussian prior and Gaussian noise with known variance, similar to that studied in [5]. More

formally, let $\tilde{\theta}_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ drawn from the prior on theta, and at each time t, we make the following updates on μ_t and Σ_t given that we take action $a_t \in \mathbb{R}^d$ and observe reward $r_t = \theta^T a_t + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ and θ is drawn from the prior $\mathcal{N}(\mu_0, \Sigma_0)$. The updates of the parameters are as follows:

$$\Sigma_t = \left(\Sigma_{t-1}^{-1} + \frac{a_t a_t^T}{\sigma^2}\right)^{-1} \qquad \hat{\theta}_t = \Sigma_t \left(\Sigma_{t-1}^{-1}\hat{\theta}_{t-1} + a_t r_t/\sigma^2\right)$$

The derivation for these posteriors can be found in the Appendix. Thompson Sampling has an adaptation that can be perceived analogous to that of UCBG: among the available arms choose a set of $T - t + 1$ number of arms and choose one among them uniformly at random. In Thompson Sampling, however, this approach corresponds to choosing an arm, uniformly at random, among the closest $T - t + 1$ arms to the θ_t^{TS} sampled from the posterior.

2.5.2 Results

We run 2 sets of experiments with a modification on the reward function: when the reward function has an underlying Gaussian distribution and when it has Bernoulli distribution. Both means are, naturally, M_{ij}, where M is the resulting completed matrix as explained in the beginning of this section.

1. Gaussian Reward: The reason we work with Gaussian reward distribution, even though it is more realistic that the matching rewards come from Bernoulli distribution in the online setting, is that we know how to work with a Gaussian prior in Thompson Sampling. The posterior is calculated as given in the previous subsection. The methods we implement are Thompson Sampling Adaptation, Greedy, LinUCB and Alternating Heuristic. To give Thompson Sampling the benefit of the doubt, instead of using the set of U we extracted from data, we sample $\theta \in \mathbb{R}^d$ from the standard multivariate normal distribution $\mathcal{N}([0]^d, I_d)$. However, the arms $A_j \in \mathbb{R}^d, j = 1, \ldots, 5000$ are as estimated from data. So, for an arm A_j the reward we observe becomes $r(A_j) = \langle A_j, \theta \rangle + \eta_j$ where $\eta_j \sim \mathcal{N}(0, 1)$ and $\theta \sim \mathcal{N}([0]^d, I_d)$.

2. Bernoulli Reward: In Bernoulli case, the reward we observe from recommending person

j to person i has an underlying distribution of $Bernoulli(M_{ij})$ where $M_{ij} = \Theta_i^T A_j$. The methods we test are Greedy, LinUCB and Alternating Heuristic.

We run the experiments with 5000 arms and horizon $T = 50$ for the linear reward with Gaussian noise case and $T = 50, 100, 200$ for the Bernoulli reward case with instances, i.e. different θ values, and the number of instances ranges from 200 to 2500. Each instance was averaged over 10 runs. The Gaussian noise has variance 1, $\eta_t \sim \mathcal{N}(0, 1)$. The immediate pseudo regret was calculated as in Lemma 2.3.2. As seen in Figures 2-3 and 2-4, Alternating Heuristic performs better in majority of the instances.

Horizon	Greedy	LinUCB1	LinUCB2	LinUCB3	UCBG1	UCBG2	UCBG3
50	100%	95.99%	94.49%	98.39%	95.99%	93.40%	**87.54%**
100	100%	90.84%	88.92%	94.53%	87.96%	86.97%	**79.13%**
200	100%	91.95%	89.91%	102.50%	92.31%	87.94%	**82.00%**

Table 2.1: Final Regret Comparisons. For all the experiments we have, we choose the tuning parameter α for the heuristic the same as LinUCB's c. In experiments, $\alpha, c = \left(\frac{1}{2}\right)^i$ where $i = 3, 4, 5$. We report the final regret w.r.t. the Greedy method.

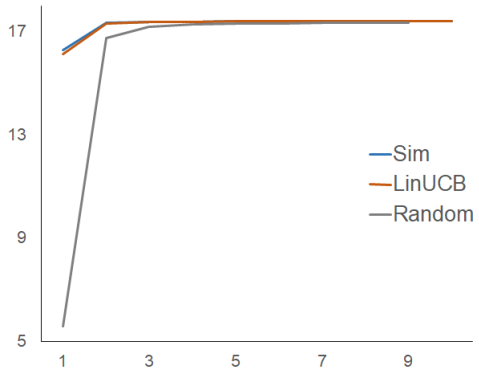
2.6 Conclusion

In this paper, we addressed online recommendations problem in which there is a budget on the number of times an item can be recommended to a user. The work is applicable in industries such as online dating platforms, subscription boxes, ad placements or streaming platforms. We investigate the most general version for which this budget is strictly 1, making any recommendation-user pair non-repeatable. In the model we worked on, the reward function is assumed to be linear and the idea is to estimate the user preferences. We give an ideal algorithm that generalizes LinUCB, however, it is computationally hard to implement this generalized algorithm. Instead, stemming from that, we give an alternating heuristic that depends on the similarities of arms (items). In the experiments we conducted o, we see that the alternating heuristic outperforms the well-known benchmarks we tested on linear bandits applied in disposable setting.

Future work includes algorithms and experiments on the matrix version of the problem, i.e. man vs woman matrix and employ a matrix completion algorithm and extend the proposed heuristic and LinUCB of linear bandits to matrix case. In theoretical part, there is more work to be done in the approximation guarantees that the alternating heuristic might achieve. Another extension could be using side information of users while making these recommendations.

2.6.1 Appendix: Additional Experimental Results

We here present an extension to the results of the experiments that was given in Figure 2-1 and conducted for $t = 5$. Now we investigate what happens as we move along the time, i.e. during time steps $t = 100$ and $t = 195$. The setup and calculations are identical, and we test 3 different initializations and give the trend of their convergence to the final function value.



Although for $t = 5$, we observed a substantial difference between the objective function values of these initializations, in Figure 2-2, we see that the final values do not differ as much and that the number of steps it takes to converge for these initializations does not vary. We explain this twist as follows: as we move forward in time, the algorithm learns θ well enough, i.e. the confidence ball shrinks enough, such that the difference between different starting points do not differ that much. However, even though minor, we see that Similarity score initiation is still better performing initially for both $t = 100$ and $t = 195$.

In the experiments the results of which is represented in Table 2.1, we test the reward function with an underlying Bernoulli distribution. This is a more realistic scenario, if we consider the dating data from a higher perspective, matrix has 1's and 0's for people having matched or unmatched respectively. Therefore, Bernoulli is a natural way to model matching probabilities of users in these platforms. In the results, we give the comparison proportional to Greedy. In all the instances, we used the same tuning parameter of LinUCB with the similarity score's parameter for fairness. All 3 of which we can see that, the Alternating Heuristic performs better than the other instances.

Along with the above experiments, we also give the average percentages with respect to

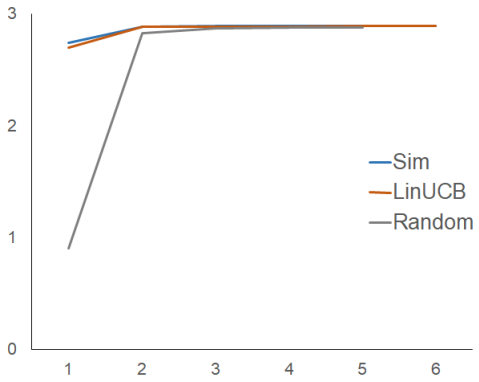


Figure 2-2: Objective function vs. iteration number. Performance of alternating heuristic with three different initialization. Results are averaged over 200 instances. *(Left):* Observed when t = 100 . *(Right):* Observed when t = 195.

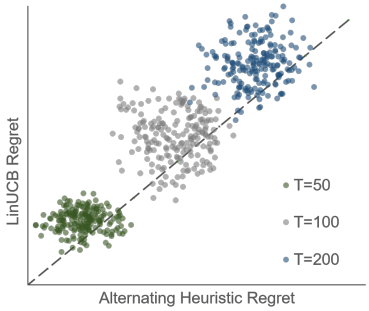


Figure 2-3: Alternating Heuristic vs LinUCB final regret. We report 200 instances for each T with the best parameter for LinUCB.

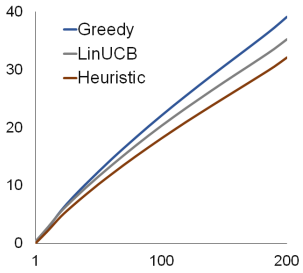


Figure 2-4: Regret vs Time. The best parameter for LinUCB and the corresponding Heuristic parameter was chosen.

different horizons and different reward functions, i.e. Gaussian and Bernoulli.

Horizon	Greedy	LinUCB1	UCBG1	LinUCB2	UCBG2	LinUCB3	LinUCB3	TS-Adaptation
50	100%	93.29%	85.66%	89.82%	82.56%	96.56%	**79.49%**	109%

Table 2.2: Final Regret Comparisons for linear mean reward with Gaussian noise

In the experiment, the results of which we present in Table 2.2, the reward function is Gaussian with mean M_{ij} and variance 1. The reasoning behind using Gaussian noise is that we know how to work with the posteriors of Thompson Sampling when the prior we have is Gaussian. Also, the adaptation was an easily implementable one. Surprisingly, however, Thompson Sampling adaptation had a higher regret averaged over the instances than that of

greedy policy. We do not know why Thompson Sampling would behave differently, it could be the case that disposable setting has affected the algorithm's performance. This is subject to further analysis and investigation in future work.

It is observed from the experiments conducted, even though we do not perfectly tune the Alternating Heuristic's similarity parameter, that the heuristic performs significantly better than the major benchmarks of linear bandits.

Chapter 3

Meta Linear Bandits

3.1 Introduction

Stochastic linear bandits is an online learning model that associates each action with a feature vector and assumes the mean reward is the inner product between the feature vector and an unknown parameter vector [1], [24], [25]. There are numerous applications that make use of this framework, such as serial webpage recommendations, pricing, or various other problems about modeling user behavior. Generally speaking, the idea is to learn these unknown parameters of users within a time frame called *horizon*.

Previous work on linear bandits has focused primarily on the case where the horizon is much larger than the dimension of the space. The main reason for this is that the minimax lower bounds are linear as opposed to sublinear in horizon in the worst case when the opposite is true, and the desired rate of learning may not happen in an adversarial setting. However, many real world applications deal with high dimensional settings in which the horizon may not be as large as these algorithms require. The framework we propose is what sheds some light into this problem: we can learn from the combination of different games where the dimension is much larger and the underlying structure of these distinct parameters is low-rank. In other words, on a higher level we also learn about the lower dimensional space from which the parameters come from. We place this idea in the meta learning with bandits literature.

Consider an online system where we see a series of users coming in, staying for a fixed amount of rounds, receiving recommendations, observing rewards and leaving. The platform

can consider the rewards as implicit, such as the time spent, or explicit such as the binary 'click-no click' proxy measure. The problem the platform is concerned with is that, given that these users have unknown feature vectors coming from a lower dimensional subspace, what kind of approach can we take to learn more and/or better about the current and future users so as to maximize the cumulative reward? The problem reveals itself to be 2-fold: the individual parameter learning part, which is what most of bandit literature is focused on, and the meta learning part on a higher layer, which is learning the underlying structure these parameters come from.

Paper	Bandit Type	Meta Regime	Assumptions
[39]	Linear bandits	Sparsity	High dimensional
[21]	Linear Bandits	Bias	Common distribution
[8]	Linear Bandits	Distribution prior	Knowledge of Prior
This paper	Linear Bandits	Low dimensional subspace	High dimensional

Meta bandits are a part of meta learning, the 'learning-to-learn' setting where the agent has extra information on the structure behind the incoming parameters to the system and learns the parameters to that of the structure as well. In the bandit world, the algorithms also try to learn the underlying information about these parameters so as to use it to discover the parameters themselves. We see different regimes of meta learning within bandits literature such as models with a prior knowledge on the distribution over these parameters [21], or over the arms [49], meta learning over different bandit models [83], sparsity [39], low rank reward matrix where the parameters are tensors [57]. We give a more structured explanation in the above table. Our work fills the gap where the parameters come from an unknown lower dimensional subspace and the dimension we operate on can be much larger than the horizon.

In meta learning, we also define an *oracle* with the knowledge of this underlying structure. For instance, if these parameters come from a common distribution, then the oracle has the exact knowledge on the parameters of this distribution. In a sparse setting, it is the knowledge on which dimensions are active dimensions. In our case, it is the knowledge of the lower dimensional space, or more formally, the knowledge of the spanning vectors which span the $s-$dimensional linear subspace. The meta level algorithm performances are compared to that of *oracle* which acts as the baseline.

Our Contributions Our work fills in the gap in the literature where we see a high dimensional multi-episode setting with a meta level regime of low rank design. We highlight the main contributions below.

1. Algorithm: Having introduced the new concept of meta linear bandits in high dimensional regime, we propose an algorithm Projected LinUCB that relies on *greedy* projections on the meta level, and lower dimensional optimistic UCB-based algorithms on the episodic level.

2. Theory: We give a lower bound of $\Omega(d + \sqrt{sdNT} + Ns\sqrt{T})$ based on a special case: sparse linear bandits. We further give upper bounds on the algorithm $\tilde{O}(d + s\sqrt{dNT})$.

3. Experiments: We conduct experiments on both synthetic and real world data, and see that compared to other regimes, our method outperforms well known algorithms in current literature.

The algorithm we propose does not use an optimism based principle on meta level. Instead, it utilizes a greedy approach. The interesting part is that the drawbacks of greedy algorithms, such as getting stuck in a suboptimal solution and over-exploit while under exploring, does not apply to the low rank setting we work on. The reason is that, the meta level exploration still exists even though the algorithm is greedy, through the noise on the episodic level and the way that meta level approximation works: re-estimating the parameters in $d-$dimensions.

3.1.1 Previous Work

The major category this work falls under is bandit algorithms. For a more fundamental and extensive bandit and linear bandit review, see [53], [24].

Classic bandits in meta setting have been studied in [83], [77]. Both of these papers are concerned about the bandit algorithm selection part as the meta-learning problem and use bandits on the algorithm selection problem on top. The classical multi-armed bandit problem, the problem where a player has to make sequential decisions about which action to take given the past observations, was introduced by [64]. The problem is well studied for a couple of decades now, pioneered and put together by [35], who introduced the all-time famous *Gittins*

Index. The known best algorithm for the classic simple MAB is the *Upper Confidence Bound (UCB)* algorithm by [48].

In the intersection of recommender systems and MAB literature, a natural approach is that the items to be recommended are modeled as arms of the bandits and each user is the player. How to use certain MAB techniques for recommendations are studied before, such as in [16], [80]. In recommendations setting, however, learning a user's preference is most sought-after, so linear bandits, first introduced in [6], is an intuitively better method to capture the multi-dimensional relationship between the items and the users. The flavor of linear bandits is different from the classic MAB in the sense that rather than trying to estimate the arms with no prior information, we try to learn the player's characteristic vector by trying out new arms whose information is available. It is called linear MAB since the reward function is linear in player's feature vector. Linear multi-armed bandits have their own UCB algorithm with a similar logic to that of simple bandits, which is called LinUCB, LinReL, or OFUL. We call the algorithm LinUCB throughout this paper, and is explained more in detail in Section 2.3.

In terms of low rank structure among linear bandits, to the best of our knowledge, [49] is the only paper that introduces a low-rank structure behind the noisy arms observed throughout the linear bandit game. However, there is no meta setting that a learning happens and the only uncertainty that they introduce comes from the observed arms being multidimensional random variables. Another low-rank linear bandit structure studied in literature is [57]. Their work is quite different from ours, due to the matrix structure of theirs, and having low rank structure for the single matrix Θ, whereas we are concerned with a more sequential game where the linear bandits are defined in a $d-$dimensional space, rather than a matrix one. For example, [37] studied low rank bandits with latent structures using robust tensor power method. [50] imposed low-rank assumptions on the feature vectors to reduce the effective dimension.

Sparsity is a special case of low rank structure and sparse linear bandits have been studied in different settings. Linear bandits with sparsity in [53] does not have a high dimensional regime. High dimensional sparse linear bandits have been studied in [39], but they do not cover the general case of lower dimensional structure. We also look at a meta regime where

we observe independent games but the parameters are correlated.

A more relevant model recently introduced is [21], where the authors analyze the case where an underlying distribution exists behind the θ^n regime, and while recovering the true θ, they also recover the underlying distribution. What we introduce and argue is a better suited model is the underlying regime having a lower dimensional structure that we do not observe.

Another important work is [58], [82], where they consider a low rank model only on the reward matrix onlys, however, we are concerned with the low-rank model among θs.

3.1.2 Notation

Let $A \subset \mathbb{R}^d$ denote the set of vectors in $d-$dimensional space, where $a_1, \ldots, a_K \subset A$ are the available actions to the agent. We assume that the arm-set spans the entire $\mathbb{R}^d$, for, if not, one can always project arms onto the lower dimensional space and play the game in that where the set spans the entire space. Let e_i denote the unit vector for which the ith dimension is 1 and the rest of the dimensions are 0. If k is a positive integer, we use the notation $[k] = \{1, \ldots, k\}$. We define $\lambda_s(M)$ as the s^{th} largest eigen value of matrix M, and $\lambda^-(M)$ and $\lambda^+(M)$ the smallest and largest *positive* eigenvalues of M, respectively. We denote the $s-$dimensional identity matrix as I_s and $M^\dagger$ denotes the *pseudo inverse* of matrix M. $\|v\|_M$ denotes the $M-$matrix norm of a vector and is defined as: $\|v\|_M = \sqrt{v^T M v}$.

3.2 Model

We start by re-introducing the stochastic linear bandits framework for completeness. The notation introduced will be used throughout the paper.

3.2.1 Stochastic Linear Bandits

The standard linear bandit problem is the problem that a set of $d-$dimensional arms are revealed to the agent, where the agent's goal is to maximize the expected cumulative reward in this $T-$round online game. The agent at each $t \in [T]$ tries to estimate a parameter $\theta^* \in \mathbb{R}^d$ as accurate as possible by immediately observing the rewards. Rewards are bilinear in feature

vectors with additive sub-Gaussian noise, i.e.

$$r(a) = \langle a, \theta \rangle + \eta$$

Assumption 3.2.1. The following are the common assumptions in literature.

1. Bounded arms: $\|a\| \leqslant L$ for all $a \in A$
2. Bounded θ: $\|\theta^*\| \leqslant S$
3. Sub-Gaussian noise: $\|\eta\|_{\psi_2} \leqslant \sigma$
4. Finite arm set: $A = \{a_1, \ldots, a_K\}$
5. Bounded rewards: $\langle a, \theta^* \rangle \in [-1, 1]$

The goal of the agent is to maximize cumulative reward, or better yet, minimize a metric called *regret*. Let r_t denote the immediate regret given as

$$r_t = \langle a^* - a_t, \theta^* \rangle \tag{3.1}$$

where a^* is the arm that maximizes the expected reward, formally

$$a^* = argmax_{a \in A} \langle a, \theta^* \rangle \tag{3.2}$$

LinUCB Algorithm The standard stochastic linear bandit problem has been studied extensively, and has various order optimal solutions up-to logarithmic factors. A well-known algorithm is called LinUCB and/or OFUL in different works [2] , [25]. Below we give an overview of the practically identical methodology studied in these, and what our algorithm will eventually largely draw from.

Confidence Sets The LinUCB algorithm relies on using previously-observed rewards to construct carefully tuned confidence sets on θ^*. The following construction is based on that of [1]: Let $\lambda > 0$ be an arbitrary constant (in practice, this can be treated as a tuning

parameter). Suppose at the beginning of time $t+1$, the policy at hand has selected arms $a_1, \ldots, a_t$ and observed rewards $r_1, \ldots, r_t$. We construct the matrix Σ_t as

$$\Sigma_t = \sum_{i=1}^{t} a_i {a_i}^\top + \lambda I,$$

where I is the identity matrix. Then letting $\hat{\theta}_t$ denote the regularized least-squares estimator of θ^*, we have

$$\hat{\theta}_t \equiv \Sigma_t^{-1} \sum_{i=1}^{t} a_i r_i,$$

and finally, the confidence set used in the LinUCB algorithm is the following ellipsoid:

$$C_t = \{\theta \in \mathbb{R}^d : \|\theta - \hat{\theta}_t\|_{\Sigma_{t-1}} \leqslant \beta_t\}, \tag{3.3}$$

where

$$\beta_t = \sigma \sqrt{2 \log T + d \log \frac{d\lambda + TL^2}{d\lambda}} + \sqrt{\lambda} S. \tag{3.4}$$

The LinUCB algorithm, then, selects the arm that has the highest "potential" reward over all $\theta \in C_t$, i.e. it selects

$$\underset{a \in A}{\operatorname{argmax}} \max_{\theta \in C_t} \langle a, \theta \rangle.$$

The ellipsoidal form of C_t enables a closed-form expression for the inner maximization above:

$$UCB_t(a) \equiv \max_{\theta \in C_t} \langle a, \theta \rangle = \langle a, \hat{\theta}_t \rangle + \beta_t \|a\|_{V_t^{-1}}. \tag{3.5}$$

The algorithm has a regret upper bound of $\tilde{O}(d\sqrt{T})$, where $\tilde{O}$ hides logarithmic terms. LinUCB is order optimal, since the lower bound on regret of stochastic linear bandits is $\Omega(d\sqrt{T})$.

3.2.2 Meta Linear Bandits

We introduce a new concept we call high dimensional meta linear bandits where we observe not a single, but $N-$many different parameters sequentially in a high dimensional space. Specifically, let n denote the episode at the order of which the current parameter arrived.

Then for each episode $n \in [N]$, we have T rounds of action taking. We are interested in the high dimensional setting, which is case when the dimension can be much higher than the horizon, i.e. $d \gg T$. We further require $s \ll T$, due to the necessity that we need at least s observations in a full-rank setting of rank s to be able to make a plausible estimation. Let θ^n denote the true parameter at episode n for which the following reward function is valid:

$$r^n(a) = \langle a, \theta^n \rangle + \eta \tag{3.6}$$

where η is a σ sub-Gaussian noise, and $a \in A^n$ where A^n is the decision set at episode n. The sets A^n for $n \in [N]$ may differ among episodes, however, for simplicity, we will assume that the arm-sets are equivalent across episodes. Hence, we will denote it as a single set A. Note that all theorems and assumptions can be easily extended to a varying decision set regime. Furthermore, we will work with Gaussian noise for simplicity, however, the results can be extended to sub-Gaussian case easily.

Low Rank Setting We are interested in the setting where the parameters come from a lower dimensional subspace. Define $\Theta^n \in \mathbb{R}^{d \times d}$ to be the following matrix:

$$\Theta^n = \frac{1}{n} \sum_{j=1}^{n} \theta^j (\theta^j)^\top.$$

In our high-dimensional low-rank setting, this matrix has rank s, where $s \ll d$. We denote the singular value decomposition (SVD) of this matrix as

$$\Theta^n = V \Lambda V^\top. \tag{3.7}$$

We also define the projection matrix: $P = VV^\top$. Hence for a given θ^n, we have $P\theta^n = \theta^n$. Note that $V\theta \in \mathbb{R}^s$ is going to give the coordinates of θ in the lower dimensional subspace. This will be used frequently in the algorithms to be discussed in the following sections.

Oracle Oracle in meta learning problem is defined as the setting in which the agent has the full knowledge of the underlying space, i.e. true $V-$matrix satisfying $VV^\top = P$. Then, the

oracle may execute one of the well-known order-optimal algorithms, i.e. LinUCB, Thompson Sampling etc. This concept matters when we compare the subspace recovery precision of the algorithms at hand to that of a baseline.

Regret: The metric we wish to optimize for is *regret*. The regret definition is 2-fold. We analyze a higher layer meta regret as well as a within-episode episodic regret.

- Meta Regret: This is the regret we observe due to the lack of knowledge in $V-$space, the $s-$dimensional space spanned by row vectors of V. It can be interpreted as the performance difference in scale between the algorithm at hand and an order-optimal algorithm with the knowledge of this $V-$space. We denote the meta regret as $\tilde{R}_T^n$ for an episode n.

- Episodic Regret: This is the regret we observe due to lack of knowledge on θ^*. It is the standard regret definition within a $T-$dimensional game. Formally, at episode n we have

$$R^n = \sum_{t=1}^{T} (\theta^n)^\top (a^n)^* - (\theta^n)^\top a_t \tag{3.8}$$

  where $(a^n)^*$ is the arm that maximizes the expected reward with θ^n.

3.2.3 Lower Bounds

The lower bounds we provide are based on a special case of meta linear bandits: sparse linear bandits. Consider the high dimensional meta linear bandits case for which, w.l.o.g,

$$P = \begin{pmatrix} I_s & 0 \\ 0 & 0 \end{pmatrix}$$

Then, the problem is similar to that of the sparse linear bandits where the game lasts for NT rounds. The following theorem follows from lower bounds derived for the bandits with sparsity [53].

Theorem 3.2.2. *For any policy, there exists a sequence of parameters* $\theta^1, \ldots, \theta^N$ *such that*

$$\sum_{n=1}^{N} R_T^n = \Omega(d + \sqrt{sdTN} + Ns\sqrt{T}) \tag{3.9}$$

Proof Sketch. The three terms that appear in lower bound come from three distinct constructions:

1. The initial term d in the above bound comes from any algorithm having to explore d dimensions and observe linear regret in d due to high dimensional regime.

2. The second term comes from a sequence of orthogonal θ_js where $T = \frac{TN}{s}$, $N = s$, and hence applying the sparse linear bandit lower bound, we derive this term.

3. The third term comes from the best case scenario, i.e. oracle, where we know the true V, we observe $Ns\sqrt{T}$ regret due to independent $N-$linear bandits.

Combining these 3 points the bound follows. The details can be found in appendix. ■

The lower bound is crucial in the episodic regret level. Notice that, on average, within an episode we see the first two terms disappearing as N goes to infinity. The third term is due to the lack of knowledge on true θ^n and it comes from the classic minimax lower bounds on the linear bandits literature, and observed regardless of the knowledge on V.

3.3 Algorithm & Bounds

We now propose an algorithm to solve the high dimensional meta linear bandit problem: Projected LinUCB. As one might guess the algorithm operates in two stages: meta level and episodic level. In meta level, the goal is to estimate an $s-$dimensional subspace to project the arms onto and work through the next episode from there. In the episodic level, the classic linear bandit problem is being solved, but in $s-$dimensional estimated space. The general idea is to operate on an approximate lower dimensional setting so as to avoid the drawbacks from the high dimensional setting.

$X_t^n \in \mathbb{R}^{d\times t}$ denotes the matrix whose columns are the arms played up until time t in episode n. Σ_t^n is the regularized matrix in the form $\Sigma_t^n = \lambda I + X_t^n (X_t^n)^\top$ where λ is a

regularization parameter. The main method is that after each episode ends, we re-calculate the matrix $\hat{V}$ using the previously observed data and calculated $\hat{\theta}$. Note that at the beginning of the game, since $T \ll d$, we do require a certain number of episodes, $\tilde{n}$, dedicated to exploration. During these episodes, the algorithm picks arms orthogonal to each other so as to maximize the information gain from them. Namely, let $X^n \in \mathbb{R}^{T \times d}$ be the arm matrix whose rows are the arms played up until the end at episode n.The algorithm then is to pick arms at each episode n such that the arms played are unit vectors of $d-$dimensional space: $e_i \in \mathbb{R}^d$ for all $i \in [d]$.

First, for each of these θ^n, we create a $d-$dimensional Ridge estimator $\hat{\theta}^n$ such that:

$$\hat{\theta}_T^n = (\Sigma_T^n)^{-1}(X_T^n)^\top Y_T^n$$

where $Y_T^n \in \mathbb{R}^T$ is the noisy reward observed throughout episode n for T rounds.

Below in Algorithm 3, we provide LinUCB algorithm on a lower dimensional setting, where we require a V_s-matrix to switch to a lower dimensional setting. The algorithm has the same logic explained in Section 4.3.1, except that it projects the high dimensional arms onto the given subspace at the beginning of the game using matrix V_s, and works with the projected armset $\tilde{A}$ throughout T rounds.

Subspace Estimation The main component of the Projected LiNUCB algorithm is the subspace estimation on meta level. After collecting a set of $\hat{\theta}$ estimates for $\tilde{n}$ episodes, the algorithm creates the following matrix estimate of Θ:

$$\hat{\Theta} = \frac{1}{n}\sum_{j=1}^{n} \hat{\theta}^j (\hat{\theta}^j)^\top \tag{3.10}$$

Then, it takes the first s principal dimensions of this matrix by performing an SVD, and calculate a projection matrix $\hat{P}$. We have:

$$\hat{\Theta} = \hat{V}\hat{\Lambda}\hat{V}^\top \tag{3.11}$$

Then, we take the first s eigenvectors of $\hat{V}$: $\hat{V}_s$, which will act as the vessel to the lower dimensional space we will play the LinUCB-s game on. Then, we feed the arm set A and $\hat{V}_s$ along with the tuning parameters to LinUCB-s and receive a matrix X_T and a reward vector Y_T. After this, we generate a $d-$dimensional $\hat{\theta}$ using the ridge estimator and update $\hat{\Theta}$ to run SVD on and retrieve a better estimate of $\hat{V}$, and repeat through episodes.

Exploration on Meta Level The surprising part of the algorithm is that, even though on the meta level we work with a greedy-type algorithm, i.e. pick the best fit projection, there is an exploration component at play in disguise. The gist of it is that after playing the game in $s-$dimensional space spanned by $\hat{V}$, we re-calculate $\hat{\theta}$ in $d-$dimensional space, with $d-$dimensional arms. Note that the estimate $\hat{\theta} \in \mathbb{R}^d$ does not necessarily lie in $span(\hat{V})$. Through the underlying reason that the rewards come from the $s-$dimensional space spanned by V, we introduce an exploratory nature to the meta level algorithm as well.

3.3.1 Upper Bound on Regret

We provide bounds on 2 types of regret: *episodic* and *meta*. As explained in model section, episodic regret is the regret we observe during one $T-$round episode, where we compare a policy to the optimal one. Meta regret, on the higher level, compares the algorithm with that with the knowledge of the subspace, i.e. oracle that knows the true V-matrix. The following theorem demonstrates upper bounds on the meta and episodic regrets of Projected LinUCB.

Theorem 3.3.1. *Suppose the assumptions given in Assumption 4.3.1 hold. Further assume that there exists an n for which $\lambda_s(\Theta^n) = \Omega(\sqrt{\frac{\log d}{n}})$ and $n \geqslant d/T$. Then for such n, with probability $1 - 3\delta$, for $\delta = O(1/T)$, the following cumulative regret bound holds for episodic regret at episode $n+1$ after T rounds:*

$$R_T^{n+1} = \tilde{O}\left(d + s\sqrt{T} + s\sqrt{\frac{T}{n}}\right)$$

Furthermore, for $N-$episode meta regret, we have

$$\sum_{n=1}^{N} \tilde{R}_T^n = \tilde{O}(s\sqrt{NT})$$

Proof Sketch. The proof is similar to that of linear bandits. However, we observe additive components in both confidence set construction and cumulative regret bounds that depend on the subspace estimation error. We give an overall interpretation of the theorem proof below.

Projection Error Here we establish the error bound on the $2-$norm of the difference between the estimated and actual projection matrices.

Theorem 3.3.2. *Suppose $\lambda_s(\Theta^n) = \Omega(\sqrt{\frac{\log d}{n}})$ and $n \geqslant d/T$, where Ω hides terms dependent on L, σ, λ and S. Then with probability $1 - 2\delta$,*

$$\|P - \hat{P}\| = O(\frac{1}{\sqrt{n}}\sqrt{\log \frac{2d}{\delta}}) \tag{3.12}$$

The detailed proof of the above theorem can be found in the appendix. The proof mainly utilizes Davis-Kahan's $sin\Theta$ theorem [78], and the fact that for the first d/T episodes, the algorithm only plays the unit vectors to maximize informational gain. The term $\log d$ comes from the high probability matrix concentration bounds. The requirement on the $\lambda_s(\Theta)$ is a necessary one to be able to recover the final space. Once we analyze the extreme case it is more clear why: imagine $\theta^1 = \theta^2 = \cdots = \theta^n$, and that $s \gg 1$. Then, with this knowledge, it's practically impossible to be able to recover the underlying $s-$dimensional space for the current n. This condition ensures that the true θ vectors we have observed until episode n span a large enough space with large enough eigen values, so as to be able to recover the underlying $V-$space.

Projected Confidence Sets Now we dive deep into guarantees on the confidence set we create in Algorithm 3 on $s-$dimensional estimated subspace. See the following theorem for the confidence set for θ^* at time t during some episode n.

Theorem 3.3.3. *Suppose the assumptions for standard stochastic linear bandits hold for each episode. Then with probability* $1-3\delta$*, during an episode* $n \geqslant d/T$ *at time* t*,* θ^n *lies in the set:*

$$C_s = \{\theta \in \mathbb{R}^d : \|\hat{\theta}_t - \theta\|_{\Sigma_t} \leqslant \beta_t\}$$

where

$$\beta_t = O\left(\sqrt{\log(\frac{1}{\delta}) + s\log t} + \sqrt{\frac{1}{n} st \log t \log \frac{d}{\delta}}\right) \tag{3.13}$$

where $\Sigma_t, \hat{\theta}_t$ *are as defined in the Algorithm 3 and* O *hides terms dependent on* σ, S, L, λ*.*

■

Note that the term β_t displays a familiar flavor: in classic least square confidence bounds, we have $\beta_t = O(\sqrt{\log(\frac{1}{\delta}) + s\log t})$. The additional term is produced by the error we observe through projection estimates. Notice that in the absence of an error between the projection matrices, we derive back the original confidence set for θ in $s-$dimensional space. In the final regret bound, this term disappears in episodic regret level as $n \to \infty$, however, in meta regret, it accumulates to a $\sqrt{N}$-order regret.

3.3.2 Discussion

The episodic regret contains a term proportionate to d, which comes from the setting that for the first d/T episodes where the algorithm randomly explores by playing orthogonal arms. Note that this term is inevitable for any algorithm since learning requires at least $\Omega(d)$ exploration. The second term, $s\sqrt{T}$ is the regret we observe due to the lack of θ^* knowledge. Notice that it is the only term we have for oracle regret. The last term $s\sqrt{\frac{T}{n}}$, is due to the projection error. The important observation here is that as $n \to \infty$, the third term disappears as we have a close-to-optimal estimate of the underlying V in the meta problem. As per meta regret, we are interested in the cumulative regret we observe due to the lack of knowledge in V. The meta regret is calculated by summing the term $s\sqrt{\frac{T}{n}}$ over all N. It basically says that in the meta level, the error we accumulate is in the order of $\sqrt{N}$, which matches the lower bound provided in the previous section.

3.4 Experiments

Synthetic Experiments In this section we provide a set of experiments where we compare the cumulative regret across episodes. Notice that regret can have 2 different meanings: regret with respect to oracle and regret with respect to the optimal policy, as is analyzed in Section 3.3. We will compare policies based on both perspectives.

Oracle As mentioned in Section 3.2.2, the oracle is the policy for which the underlying subspace, or in other words the matrix V, is known. We argue that the meta regret cannot have a clearly defined immediate regret as there are multiple order-optimal policies and there is a randomness as to which arm is the arm that the oracle would play. However, we can still look at the cumulative reward difference between policies and assess how *close* the algorithms at hand are to oracle.

Mainly, the oracle first projects all arms in set A onto the known $s-$dimensional subspace, i.e. $\tilde{A} = \{Va | a \in A\}$. After this operation, notice that $\tilde{A}$ has $s-$dimensional arms. Now, for experimental purposes, we picked LinUCB as the algorithm to be applied: pick the arm that maximizes $\langle a, \hat{\theta}_t \rangle + \gamma \|a\|_{\Sigma_t^{-1}}$, where γ is a tuning parameter and $a \in \tilde{A}$.

In experiments, we create a set of arms by randomly generating from the unit ball in d dimensions. The following step is to generate θ values in d dimensions that lie on a lower dimensional subspace. To do so, we first generated a set of θ' values in $s-$dimensional unit ball. Then, for each trial, we randomly generate a set of independent and orthogonal unit vectors in d dimensions, and use them to represent these θs in d dimensions.

K is the number of arms, N is the number of episodes and also number of different θs, and T is the horizon for each episode. Each algorithm has 2 component as one might expect: the algorithm for meta level, and the algorithm for the episodic level. The following gives the explanation as to what kind of algorithms we run for meta and episodic levels.

- Greedy+Greedy: The algorithm picks an approximate $s-$dimensional subspace in a greedy manner, and plays a greedy policy within an episode.
- Greedy + LinUCB: The algorithm picks an approximate $s-$dimensional subspace in a greedy manner, and plays an optimism based policy within an episode. This is our

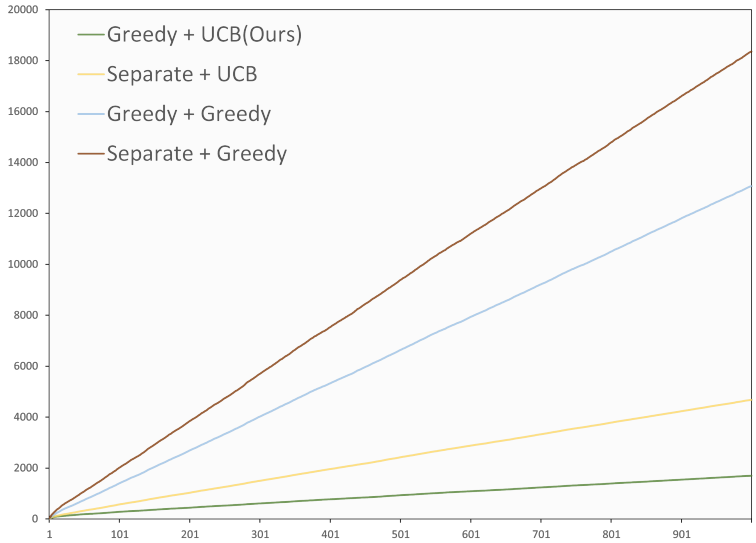


Figure 3-1: The regret vs period plot. Each point is the cumulative regret after the end of a period

algorithm that we test against other benchmarks.

- Oracle: The oracle uses the given subspace to project the game onto, and plays an optimism based policy within an episode.

- Independent+LinUCB: The algorithm plays N independent episodes, and within episode it executes an optimism based algorithm.

- Independent+Greedy: The algorithm plays N independent episodes, and within episode it executes a greedy algorithm.

The plot was created by subtracting the cumulative rewards from that of the policy *oracle*.

3.5 Conclusion

In this paper, we first introduced the model of high dimensional meta linear bandits, where the dimension d is much larger than the horizon T. We are focusing on the version for which we have N different parameters coming in to the system sequentially, where N can go to infinity, and the information we have upfront is that there is an unknown $s-$dimensional linear subspace where these N different parameters lay in: $s \ll T \ll d$. We propose a projection

based algorithm Projected LinUCB, where we play an optimism based game in the episodic level and a greedy policy on the meta level. Interestingly, we found that the greedy policy introduces a certain exploration and successfully avoids the common problem of getting stuck at a local optima.

For future work, certain assumptions can be loosened such as introducing a varying T for different games. Parallel structure is another version that could be useful in practical applications.

3.6 Appendix

3.6.1 Lower Bounds

Theorem 3.6.1. *Let $A = \{e_i \in \mathbb{R}^d\}$. For any policy π there exists a sequence of $\theta_1, \ldots, \theta_N$ such that*

$$\sum_{n=1}^{N} R_T^n = \Omega(d + \sqrt{sdTN} + Ns\sqrt{T}) \tag{3.14}$$

Proof. We make use of the following lower bound from [53]:

Theorem 3.6.2 (Theorem 24.3, [53])**.** *Assume $sd \leqslant T$ for some integer $k \geqslant 2$. Let $A = \{e_i \in \mathbb{R}^k : i \in [k]\}^s \subset \mathbb{R}^d$. Then for any policy there exists a parameter vector $\theta \in \mathbb{R}^d$ with $\|\theta\|_0 = s$ and $\|\theta\|_\infty \leqslant \sqrt{d/(sT)}$ such that $R_T(A, \theta) \geqslant \Omega(\sqrt{sdT})$.*

Now consider the oracle: the case for which we know the $V-$space. Then, by using the regular linear bandits lower bound, we have $Ns\sqrt{T}$. Finally, we take the max of these, which lead to the provided lower bound as follows.

$$\Omega(\max(d, \sqrt{sdNT}, Ns\sqrt{T})) = \Omega(d + \sqrt{sdTN} + Ns\sqrt{T})$$

■

3.6.2 Bound on Projection Error

Theorem 3.6.3. *For the P, and $\hat{P}$ defined in section 4, let $n \geqslant$, $\lambda_s \geqslant$, with probability $1-2\delta$,*

$$\|P - \hat{P}\| \leqslant \frac{2\sigma^2 \frac{\lambda^-}{(\lambda^- + \lambda)^2} \sqrt{\frac{1}{N} \log \frac{2d}{\delta}} + 2\sigma S \frac{\sqrt{\lambda^-}}{\lambda^- + \lambda} \sqrt{\frac{1}{N} \log \frac{2d}{\delta}}}{\lambda_s(\Theta) - 2\sigma^2 \frac{\lambda^-}{(\lambda^- + \lambda)^2} \sqrt{\frac{1}{N} \log \frac{2d}{\delta}} - 2\sigma S \frac{\sqrt{\lambda^-}}{\lambda^- + \lambda} \sqrt{\frac{1}{N} \log \frac{2d}{\delta}}} \tag{3.15}$$

Proof. We use Davis-Kahan theorem and Matrix Bernstein here. Define the following:

$$
\begin{aligned}
S &= \frac{1}{N}\sum_{j=1}^{N}\theta^j(\theta^j)^\top + VV^\top\Sigma_\epsilon VV^\top + VV^\top\Sigma_{\theta_\epsilon}VV^\top + VV^\top\Sigma_{\theta_\epsilon}^\top VV^\top \\
\hat{S} &= \hat{\Theta} \\
\Sigma_\epsilon &= \mathbb{E}[\epsilon\epsilon^\top] \\
&= \theta\theta^\top - 2(X^\top X+\lambda I)^{-1}X^\top X\theta\theta^\top + (X^\top X+\lambda I)^{-1}X^\top X\theta\theta^\top X^\top X(X^\top X+\lambda I)^{-1} \\
&+ (X^\top X+\lambda I)^{-1}X^\top\sigma^2 I_d X(X^\top X+\lambda I)^{-1} \\
\Sigma_{\theta_\epsilon} &= \mathbb{E}[\theta\epsilon^\top] = \theta\theta^\top - (X^\top X+\lambda I)^{-1}X^\top X\theta\theta^\top
\end{aligned}
$$

Notice now that $S \in span(V)$ and $\hat{S} \in span(\hat{V})$, hence, we can apply Davis Kahan's $sin\Theta$ theorem to bound the difference between these 2 spaces.

$$
\|\hat{P}-P\|_2 \leqslant \frac{\|\hat{\Theta}U - U\Lambda\|_2}{\delta} \leqslant \frac{\|(\hat{\Theta}-S)V\|_2}{\delta} \leqslant \frac{\|\mathbb{E}[(\hat{\Theta}-S)V]\| + \|\hat{\Theta}-S-\mathbb{E}[\hat{\Theta}-S]\|}{\delta}
$$

where $\delta = |\lambda_s(S) - \lambda_{s+1}(\hat{\Theta})|$. Using Weyl's inequality we have:

$$
\|\hat{P}-P\| \leqslant \frac{\|\mathbb{E}[(\hat{\Theta}-S)V]\| + \|\hat{\Theta}-S-\mathbb{E}[\hat{\Theta}-S]\|}{\lambda_s(S) - \|\mathbb{E}[\hat{\Theta}-S]\| - \|\hat{\Theta}-S-\mathbb{E}[\hat{\Theta}-S]\|}
$$

$$
\lambda_s(S) \geqslant \lambda_s(\Theta) + \lambda_{\min}(V^\top\Sigma_\epsilon V) + 2\lambda_{\min}(V^\top\Sigma_{\theta_\epsilon}V) \geqslant \lambda_s(\Theta) \tag{3.16}
$$

note the following based on the definitions:

$$
\begin{aligned}
\|\mathbb{E}[\hat{\Theta}-S]\| &\leqslant \|\Sigma_\epsilon\| + 2\|\Sigma_{\theta_\epsilon}\| \leqslant S(\frac{\lambda}{\lambda+\lambda^-})^2 + \sigma^2\frac{\lambda^+}{(\lambda+\lambda^+)(\lambda+\lambda^-)} + 2\frac{\lambda}{\lambda+\lambda^+} \\
\|\mathbb{E}[(\hat{\Theta}-S)V]\| &= \|\mathbb{E}[(\frac{1}{n}\sum_{j=1}^{n}(\theta_j\theta_j^\top + \theta_j\epsilon_j^\top + \epsilon_j\theta_j^\top + \epsilon_j\epsilon_j^\top) - S)V]\| = 0 \\
\hat{\Theta}-S &= \frac{1}{N}\left(\sum_{j=1}^{N}\epsilon_j\epsilon_j^\top + \sum_{j=1}^{N}\theta_j\epsilon_j^\top + \sum_{j=1}^{N}\epsilon_j\theta_j^\top - VV^\top\Sigma_\epsilon^\top VV^\top - VV^\top\Sigma_{\theta_\epsilon}VV^\top - VV^\top\Sigma_{\theta_\epsilon}^\top VV^\top\right) \\
\hat{\Theta}-S-\mathbb{E}[\hat{\Theta}-S] &= \frac{1}{n}\sum_{j=1}^{n}\left(\epsilon_j\epsilon_j^\top - \Sigma_\epsilon + \theta_j\epsilon_j^\top + \epsilon_j\theta_j^\top - \Sigma_{\theta_\epsilon} - \Sigma_{\theta_\epsilon}^\top\right)
\end{aligned} \tag{3.17}
$$

Now from Cauchy Schwartz we get the following:

$$\|\hat{\Theta} - S - \mathbb{E}[\hat{\Theta} - S]\| \leqslant \|\frac{1}{n}\sum_{j=1}^{n} \epsilon_j \epsilon_j^\top - \Sigma_\epsilon\| + \|\frac{1}{n}\sum_{j=1}^{n} \theta_j \epsilon_j^\top + \epsilon_j \theta_j^\top - \Sigma_{\theta_\epsilon} - \Sigma_{\theta_\epsilon}^\top\|$$

Now in order to bound both of these terms above we need the following theorems.

Theorem 3.6.4. *[Matrix Bernstein- SubExponential case, [71]] Consider a finite sequence $\{X_k\}$ of independent, random, self-adjoint matrices with dimension d. Assume that $\mathbb{E}[X_k] = 0$ and $\mathbb{E}[X_k^p] \leqslant \frac{p!}{2} R^{p-2} A_k^2$, and define*

$$\sigma^2 = \|\sum_k A_k^2\|$$

Then the following holds for all $t \geqslant 0$:

$$\mathbb{P}\left\{\lambda_{\max}(\sum_k X_k) \geqslant t\right\} \leqslant d \exp(\frac{-t^2}{4\sigma^2}) \tag{3.18}$$

Theorem 3.6.5. *With probability $1 - 2e^{-u}$, the following holds for an absolute constant $C > 0$:*

$$\|\frac{1}{n}\sum_j \epsilon_j \epsilon_j^\top - \Sigma_\epsilon\| \leqslant CK^2 \left(\sqrt{\frac{d+u}{n}} + \frac{d+u}{n}\right) \|\Sigma_\epsilon\| \tag{3.19}$$

Bounding Term 1 Notice that the $\epsilon_j \epsilon_j^\top$ follows a sub-Exponential distribution, hence, we can bound the first term using the concentration bound given in Theorem 3.6.5. Notice that ϵ_j follows a multivariate Gaussian distribution with variance and that the product of two sub-Gaussian random variables follow a subexponential distribution:

$$\begin{aligned} Var(\epsilon_j) &= \sigma^2 (X^\top X + \lambda I)^{-1} X^\top X (X^\top X + \lambda I)^{-1} \\ \|(\epsilon_j \epsilon_j^\top)\|_{\psi_2} &\leqslant \sigma^2 \frac{\lambda^-}{(\lambda^- + \lambda)^2} \\ \|\Sigma_\epsilon\| &\leqslant S(\frac{\lambda}{\lambda + \lambda^-})^2 + \sigma^2 \frac{\lambda^+}{(\lambda + \lambda^+)(\lambda + \lambda^-)} \end{aligned} \tag{3.20}$$

Therefore, using the above information with Theorem 3.6.5, we get

$$\mathbb{P}\left\{\|\frac{1}{n}\sum_{i=1}^{n}\epsilon\epsilon^\top - \Sigma_\epsilon\| \geqslant 2\sigma^2\frac{\lambda^-}{(\lambda^- + \lambda)^2}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}}\right\} \leqslant \delta \tag{3.21}$$

Bounding Term 2 Following from the sub-Gaussian norm established on Equation (3.20), we have the following bound

$$\mathbb{P}\left\{\|\frac{1}{n}\sum_{i=1}^{n}\theta\epsilon^\top - \Sigma_{\theta_\epsilon}\| \geqslant 2\sigma S\frac{\sqrt{\lambda^-}}{\lambda^- + \lambda}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}}\right\} \leqslant \delta \tag{3.22}$$

Combining above bounds we get, with probability $1 - 2\delta$,

$$\|P - \hat{P}\| \leqslant \frac{2\sigma^2\frac{\lambda^-}{(\lambda^-+\lambda)^2}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}} + 2\sigma S\frac{\sqrt{\lambda^-}}{\lambda^-+\lambda}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}}}{\lambda_s(\Theta) - 2\sigma^2\frac{\lambda^-}{(\lambda^-+\lambda)^2}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}} - 2\sigma S\frac{\sqrt{\lambda^-}}{\lambda^-+\lambda}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}}} = O(\frac{1}{\sqrt{N}}\sqrt{\log\frac{2d}{\delta}}) \tag{3.23}$$

where the last inequality follows from $\lambda_s(\Theta) > 2\sigma^2\frac{\lambda^-}{(\lambda^-+\lambda)^2}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}} + 2\sigma S\frac{\sqrt{\lambda^-}}{\lambda^-+\lambda}\sqrt{\frac{1}{N}\log\frac{2d}{\delta}}$, which basically translates to $n \geqslant \log dT$ for $\delta = O(1/T)$. Since we are already interested in the later game when $n > d/T$, this assumption holds naturally.

■

3.6.3 Confidence Set Construction

Theorem 3.6.6. *Suppose the assumptions above hold. Then with probability* $1 - 3\delta$, θ *lies in the set:*

$$C_s = \{\theta \in \mathbb{R}^d : \|\hat{\theta}_t - \theta\|_{A_t} \leqslant \beta_\delta\}$$

where

$$\beta_\delta = R\sqrt{2\log(\frac{1}{\delta}) + m\log(1 + \frac{tL^2}{m\lambda})} + SL\sqrt{\gamma mt}\sqrt{\log\left(1 + \frac{tL^2}{m\lambda}\right)}\|P - \hat{P}\|_2 + S\sqrt{\lambda}$$

Proof. Let $S_t := \sum_{i=1}^{t} \hat{P} X_{i-1} \eta_{i-1}$.

$$\begin{aligned}
\hat{\theta}_t &= A_t^\dagger S_t + A_t^\dagger \hat{P} \Sigma_{t-1} P \theta_* \\
&= A_t^\dagger S_t + A_t^\dagger (\hat{P}\Sigma_{t-1}(\hat{P} + P - \hat{P}) + \lambda \hat{P} - \lambda \hat{P})\theta_* \\
&= A_t^\dagger S_t + \hat{P}\theta_* + A_t^\dagger (\hat{P}\Sigma_{t-1}(P - \hat{P}))\theta_* - \lambda A_t^\dagger \theta_*
\end{aligned}$$

Now, for any $x \in \mathbb{R}^d$ we can write the following:

$$\begin{aligned}
x^\top \hat{\theta}_t - x^\top \theta_* &= x^\top A_t^\dagger S_t + x^\top A_t^\dagger (\hat{P}\Sigma_{t-1}(P - \hat{P}))\theta_* - \lambda x^\top A_t^\dagger \theta_* \\
&= \langle x, S_t \rangle_{A_t^\dagger} + \langle x, \hat{P}\Sigma_{t-1}(P - \hat{P})\theta_* \rangle_{A_t^\dagger} - \lambda \langle x, \theta_* \rangle_{A_t^\dagger}
\end{aligned}$$

Now, for $x = A_t(\hat{\theta}_t - \theta_*)$, we have the following inequality derived by using Cauchy-Schwartz inequality:

$$\begin{aligned}
|x^\top \hat{\theta}_t - x^\top \theta_*| &\leqslant \|x\|_{A_t^\dagger} \left(\|S_t\|_{A_t^\dagger} + \|\hat{P}\Sigma_{t-1}(P - \hat{P})\theta_*\|_{A_t^\dagger} + \lambda \|\theta_*\|_{A_t^\dagger} \right) \\
&\leqslant \|x\|_{A_t^\dagger} \left(\|S_t\|_{A_t^\dagger} + \|A_t^{\dagger 1/2} \hat{P}\Sigma_{t-1}(P - \hat{P})\theta_*\|_2 + \lambda \|\theta_*\|_{A_t^\dagger} \right) \\
&\leqslant \|x\|_{A_t^\dagger} (\|S_t\|_{A_t^\dagger} + \|(A_t^\dagger)^{1/2} \hat{P}\Sigma_{t-1}(P - \hat{P})\theta_*\|_2 + \sqrt{\lambda}\|\theta_*\|_2)
\end{aligned}$$

For $x = A_t(\hat{\theta}_t - \theta_*)$ we have

$$\|\hat{\theta}_t - \theta_*\|_{A_t}^2 \leqslant \|A_t(\hat{\theta}_t - \theta_*)\|_{A_t^\dagger} \left(\|S_t\|_{A_t^\dagger} + \|(A_t^\dagger)^{1/2} \hat{P}\Sigma_{t-1}\|_2 \|P - \hat{P}\|_2 \|\theta_*\|_2 + \sqrt{\lambda}\|\theta_*\|_2 \right)$$

Now we divide both sides with $\|\hat{\theta}_t - \theta_*\|_{A_t}$, which gives,

$$\|\hat{\theta}_t - \theta_*\|_{A_t} \leqslant \|S_t\|_{A_t^\dagger} + S\|(A_t^\dagger)^{1/2} \hat{P}\Sigma_{t-1}\|_2 \|P - \hat{P}\|_2 + S\sqrt{\lambda}$$

Now, we bound each term above separately.

Bound on Term 1 We use a theorem from [49]

Theorem 3.6.7 (Theorem 9, [49]). *For any $\delta > 0$, with probability $1-\delta$, for all $t \geqslant 1$,*

$$\|S_t\|^2_{A_t^\dagger} \leqslant 2R^2 \log\left(\frac{\det(B_t)^{1/2}\det(\lambda I_m)^{-1/2}}{\delta}\right)$$

Using above result and the fact that $det(B_t) \leqslant (\lambda + \frac{tL^2}{m})^m$, we get:

$$\|S_t\|_{A_t^\dagger} \leqslant R\sqrt{2\log(\frac{1}{\delta}) + m\log(1+\frac{tL^2}{m\lambda})} \tag{3.24}$$

Bound on Term 2

Lemma 3.6.8. *Assuming $\|x_t\| \leqslant L$ for all $x_t \in armset$, $\|(A_t^\dagger)^{1/2}\hat{P}\Sigma_{t-1}\|_2 \leqslant L\sqrt{t}\sqrt{\gamma m}\sqrt{\log\left(1+\frac{tL^2}{m\lambda}\right)}$.*

Proof. Recall the definition of $\Sigma_t = \sum_{i=1}^t X_i X_i^\top$. We get the bound doing the following:

$$\begin{aligned}
\|(A_t^\dagger)^{1/2}\hat{P}\Sigma_{t-1}\|_2 &= \|\sum_{i=1}^t (A_t^\dagger)^{1/2}\hat{P}X_i X_i^\top\|_2 \\
&\leqslant \sum_{i=1}^t \|(A_t^\dagger)^{1/2}\hat{P}X_i X_i^\top\|_2 \\
&\leqslant \sum_{i=1}^t \|(A_t^\dagger)^{1/2}\hat{P}X_i\|_2 \|X_i\|_2 \\
&\leqslant L\sum_{i=1}^t \|(A_t^\dagger)^{1/2}\hat{P}X_i\|_2 \\
&= L\sum_{i=1}^t \|\hat{P}X_{i-1}\|_{A_t^\dagger} \\
&= L\sum_{i=1}^t \|\hat{V}^\top X_{i-1}\|_{B_t^{-1}} \\
&\leqslant L\sqrt{t}\sqrt{\sum_{i=1}^t \|\hat{V}^\top X_{i-1}\|^2_{B_{t,i-1}^{-1}}} \leqslant L\sqrt{\gamma m t}\sqrt{\log\left(1+\frac{tL^2}{m\lambda}\right)}
\end{aligned} \tag{3.25}$$

where $\gamma = \frac{L^2}{\lambda\log(1+\frac{L^2}{\lambda})}$. Now combining above points .... , we have:

$$\|\hat{\theta}_t - \theta_*\|_{A_t} \leqslant R\sqrt{2\log(\frac{1}{\delta}) + m\log(1+\frac{tL^2}{m\lambda})} + SL\sqrt{\gamma m t}\sqrt{\log\left(1+\frac{tL^2}{m\lambda}\right)}\|P-\hat{P}\|_2 + S\sqrt{\lambda} \tag{3.26}$$

■

3.7 Upper Bound on Regret

Theorem 3.7.1. *With probability $1-3\delta$, at episode $N+1$ with T timesteps, the following regret bound holds:*

$$R_T \leqslant \tilde{O}\left(m\sqrt{T} + m\sqrt{\frac{T}{N}}\right)$$

Furthermore, for the meta regret defined in , we have

$$R_T^N = \tilde{O}(m\sqrt{NT})$$

Proof.

$$\begin{aligned} R_t &= \theta^\top a^* - \theta^\top a_t \\ &\leqslant \tilde{\theta}^\top \hat{P} a_t - \theta^\top \hat{P} a_t + 2LS\|P - \hat{P}\| \\ &= (\tilde{\theta}^\top - \theta^\top)\hat{P} a_t + 2LS\|P - \hat{P}\| \\ &\leqslant \|\tilde{\theta}^\top - \theta^\top\|_V \|\hat{P} a_t\|_{V^{-1}} + 2LS\|\hat{P} - P\|_2 \\ &\leqslant \beta\|a_t\|_{A_t^\dagger} + 2LS\|\hat{P} - P\|_2 \end{aligned} \tag{3.27}$$

Now, using the assumption of reward begin upper bounded, we write the following.

$$\begin{aligned} R_t &\leqslant 2\min\left(\beta\|a_t\|_{A_t^\dagger} + LS\|\hat{P} - P\|_2, 1\right) \\ &\leqslant 2\min\left(\beta_t\|a_t\|_{A_t^\dagger}, 1\right) + 2LS\min(\|\hat{P} - P\|, 1) \\ &\leqslant 2\beta_t \min\left(\|a_t\|_{A_t^\dagger}, 1\right) + 2LS(\|\hat{P} - P\|) \end{aligned} \tag{3.28}$$

where the last inequality comes from the assumption that we run the algorithm after the

warm up period, which gives $\|\hat{P} - P\|_2 \leqslant 1$.

$$\begin{aligned}
R_T &\leqslant \sum_{t=1}^{T} 2\beta_t \min(\|a_t\|_{A_t^\dagger}, 1) + 2LS\|\hat{P} - P\| \\
&= 2LST\|P - \hat{P}\| + 2\beta_T \sum_{t=1}^{T} \min(\|a_t\|_{A_t^\dagger}, 1) \\
&\leqslant 2LST\|P - \hat{P}\| + 2\beta_T \sqrt{T \sum_{t=1}^{T} \min(\|a_t\|^2_{\hat{V}^\top}, 1)} \\
&\leqslant 2LST\|P - \hat{P}\| + 2\beta_T \sqrt{T \sum_{t=1}^{T} \min(\|\hat{P}^\dagger a_t\|^2_{\bar{A}_t^{-1}}, 1)} \\
&\leqslant 2LST\|P - \hat{P}\| + 2\beta_T \sqrt{Tm \log \frac{\lambda m + TL^2}{m\lambda}} \\
&\leqslant C.\beta_T \sqrt{Tm \log T}
\end{aligned} \tag{3.29}$$

Since the second term is dominating, the last inequality follows. Since $\beta_t = O(\sqrt{m \log t} + \sqrt{\frac{m \log t \log d}{N}})$, as $N \to \infty$

$$R_t = \tilde{O}(m\sqrt{T})$$

Now we further prove the meta regret bound. To do so, we need to analyze the cumulative regret over N periods.

$$\begin{aligned}
\sum_{n=1}^{N} 2LST\|P - \hat{P}\| + 2\beta_T \sqrt{Tm \log \frac{\lambda m + TL^2}{m\lambda}} &\leqslant \sum_{n} 2LSTC(\frac{1}{\sqrt{n}} \sqrt{\log dT}) + C_2(m \log T\sqrt{T} + m \log T\sqrt{T \log \\
&\leqslant \tilde{O}(d + Nm\sqrt{T} + m\sqrt{NT})
\end{aligned} \tag{3.30}$$

Lemma 3.7.2 (Eliptical Potential Lemma)**.** *Let* $x_1, \ldots, x_n \in \mathbb{R}^d$, $V_t = V_0 + \sum_{s=1}^{t} x_s x_S^\top$, $t \in [n]$, $v_0 = trace(V_0)$, *and* $L \geqslant max_t \|x_t\|_2$. *Then,*

$$\sum_{t=1}^{n} \min(1, \|x_t\|^2_{V_{t-1}^{-1}}) \leqslant 2 \log \frac{\det V_n}{\det V_0} \leqslant d \log \frac{v_0 + nL^2}{d \det^{1/d} V_0}$$

■

■

Chapter 4

Bilinear Bandits with Arbitrary Arrivals

4.1 Introduction

Consider an online recommendation problem, where users arrive in a platform, such as a streaming service or media or arbitrarily, and such platform wants to make a well-informed decision on which item to show to the incoming user based on the limited data access they have. This is a common recommendation phenomenon faced by many platforms, and hence the models and algorithms to solve this problem are very widely studied. They are solved mostly by using matrix completion methods, or certain bandit algorithms.

We are focusing on the linear bandit setting where the rewards are noisy with a mean that is linear in the unknown user vectors. More precisely, stochastic linear bandits is an online learning model that associates each action with a feature vector and assumes the mean reward is the inner product between the feature vector and an unknown parameter vector [1], [24], [25]. There are numerous applications that make use of this framework, such as serial webpage recommendations, pricing, or various other problems about modeling user behavior. Generally speaking, the idea is to learn these unknown parameters of users within a time frame called horizon.

Previous work on linear bandits has focused primarily on the case where this horizon is much larger than the dimension of the space. The main reason for this is that the minimax lower bounds are linear as opposed to sublinear in horizon in the worst case when the opposite is true, and the desired rate of learning may not happen in an adversarial setting. However,

many real world applications deal with high dimensional settings in which the horizon may not be as large as these algorithms require. The framework we propose is what sheds some light into this problem: we can learn about users more efficiently across different games when the dimension is much larger and the underlying structure of these distinct parameters is low-rank. In other words, on a higher level we also learn about the lower dimensional space from which the parameters come from. We place this idea in the meta learning with bandits literature.

Meta bandits are a part of meta learning, where the agent has extra information on the structure behind the incoming parameters to the system and learns the parameters to that of the structure as well. In the bandit world, the algorithms also try to learn the underlying information about these parameters so as to use it to discover the parameters themselves. We see different regimes of meta learning within the bandit literature such as models with a prior knowledge on the distribution over these parameters [21], or over the arms [49], meta learning over different bandit models [83], sparsity [39], low rank reward matrix where the parameters are tensors [57]. Our work fills the gap where the parameters come from an unknown lower dimensional subspace and the dimension we operate on can be much larger than the horizon. The bandit model described above has strong correspondence in real-world recommender system applications. Consider an online system where we see a series of users coming in, staying for a fixed amount of rounds, receiving recommendations, observing rewards and leaving. The platform can consider the rewards as implicit, such as the time spent, or explicit such as the binary 'click-no click' proxy measure. The problem the platform is concerned with is that, given that these users have unknown feature vectors coming from a lower dimensional subspace, what kind of approach can we take to learn more and/or better about the current and future users so as to maximize the cumulative reward. The problem reveals itself to be 2-fold: the individual parameter learning part, which is what most of bandit literature is focused on, and the meta learning of the subspace on a higher layer, which is learning the underlying structure these parameters come from.

There are certain caveats in solving a cold start problem by employing a bandit algorithm, which in fact is the common and the most advanced method, is that the arm (or item in this case) knowledge come from previously made observations, i.e. old data. Although matrix

completion methods are quite pristine, once the latent space changes from dataset to dataset, the knowledge transfer does suffer from some loss. We are trying to solve a particular type of problem caused by this: arms have high dimensional estimated vectors, or the arms are new themselves which would result in a high dimensional setting, and that the true dimension is much lower than the ambient dimension.

The problem we are concerned with falls under a more realistic setting where $d \gg T^j$, where T^j is the amount of times a user j has appeared in the system. In a high dimensional setting, well known classical algorithms do not 'start working' until a certain amount of observations have been made, and play a randomly selected arm (or show a randomly selected item in this context), which hurt the overall performance of the recommender system.

To overcome this issue, we require a further common assumption: *the user vectors come from a lower dimensional subspace*. Notice that this assumption is not a newly-made one, since the entire matrix completion literature feeds from the assumption that the reward matrix has low rank structure. This, consequently, means the matrix whose rows are the user vectors in the system, has a low rank structure as well.

Contributions The contributions we make in this paper is 3-fold:

1. Model: We model this problem by using high dimensional meta linear bandits with arbitrary arrivals. The $d-$dimensional user vectors lay in a lower $s-$dimensional subspace. The rewards are bilinear in these unknown user vectors.

2. Algorithm: We propose a novel iterative algorithm based on a convex relaxation of a rank minimization problem. We then solve this problem by approximating it using SoftImpute, and further borrow from optimism based principles from matrix completion literature.

3. Experiments: We test our algorithm against well known methods from literature on a real-world data set from NetEase music streaming platform, containing over 50 million impressions. Our algorithm outperforms the benchmarks on various settings we test on. We further analyze the effect of the relationship between the ambient dimension and underlying rank.

4.2 Previous Work

We have given some background on the previous work on bandits in the previous section. There exist a few studies on pulling a pair of arms as a unit action. [47] consider the k-armed bandit with N_1 left arms and N_2 right arms. The expected rewards can be represented as a matrix $N_1 \times N_2$ where the authors assume it has rank r. The main difference from our setting is that they do not assume that the arm features are available, so our work is related to [47] in the same way as the linear bandits are related to K-armed bandits. The problem considered in [42] is essentially a rank-one version of [47], which is motivated by a click-feedback model called position-based model with N_1 items and N_2 positions. This work is further extended to have a tighter KL-based bound by [42]. [84] propose a more generic problem called factored bandits whose action set is a product of atomic L action sets rather than two. While they achieve generality by not require to know the explicit reward model, factored bandits do not leverage the known arm features nor the low-rank structure, resulting in large regret in our problem. There are other works that exploit the low-rank structure of the reward matrix, although the action is just a single arm pull. [67] consider the contextual bandit setting where there are discrete contexts and arms, but do not take into account the observed features of contexts or arms. [38] consider a similar setting, but employ the robust tensor power method for recovery. [43] study essentially the same problem, but make assumptions on the prior that generates the unknown matrix and perform online matrix factorization with particle filtering to leverage the low-rank structure. There has been a considerable amount of contextual bandit studies that exploit other structures, where the context is usually the user identity or features. For example, [33], [32], leverage the clustering structure of the contexts. In[22] and [73], a graph structure of the users is leveraged to enjoy regret bound that is lower than running bandits on each context (i.e., user) independently. [28] introduce a multitask learning view and exploit arm similarity information via kernels, but their regret guarantee is valid only when the similarity is known ahead of time. In this vein, if we think of the right arm set Z as tasks, we effectively assume different parameters for each task but with a low-rank structure. That is, the parameters can be written as a linear combination of a few hidden factors, which are estimated on the fly rather than being

known in advance. [40] consider low-rank structured bandits but in a different setup. Their reward model has expected reward of the form in our setting, they consider a continuous arm set only, so their algorithm cannot be applied to our problem. Our subroutine LowOFUL is quite similar to SpectralUCB of [46], which is designed specifically for graph-structured arms in which expected rewards of the two arms are close to each other when there is an edge between them. [39] study a similar regularizer in the context of sparse linear bandits under the assumption that a superset of the sparse locations is known ahead of time. [79] consider a setup where they assume an estimate of the subspace is available, but their regret bound still depends on the total dimension p. Sparsity is a special case of low rank structure and has been studied in different settings. Linear bandits with sparsity in [53] does not have a high dimensional regime. High dimensional sparse linear bandits have been studied in [39], but they do not cover the general case of lower dimensional structure. We also look at a meta regime where we observe independent games but the parameters are correlated.

For matrix completion methods, there is a vast sea of work to look at, [19], [44], [17], [63], [18] . There are 2 main differences between matrix completion and what we want to solve: the entries are not independently observed, the rows to be observed at each time step is arbitrary, and the column vectors are known.

A more relevant model recently introduced is [21], where the authors analyze the case where an underlying distribution exists behind the θ^n regime, and while recovering the true θ, they also recover the underlying distribution. What we introduce and argue is a better suited model is the underlying regime having a lower dimensional structure that we do not observe. Another very relevant work is [41], where the agent can pull an arm that has a bilinear matrix structure, i.e. pull the entries. This model is different than ours, since the agent is allowed to pull any entry, as we work on an arbitrary row arrival.

4.2.1 Notation

Let $A \subset \mathbb{R}^d$ denote the set of vectors in $d-$dimensional space, where $a_1, \ldots, a_K \subset A$ are the available actions to the agent. We assume that the arm-set spans the entire $\mathbb{R}^d$, for, if not, one can always project arms onto the lower dimensional space and play the game in that where the set spans the entire space. Let e_i denote the unit vector for which the

ith dimension is 1 and the rest of the dimensions are 0. If k is a positive integer, we use the notation $[k] = \{1, \ldots, k\}$. We define $\lambda_s(M)$ as the s^{th} largest eigen value of matrix M, and $\lambda^-(M)$ and $\lambda^+(M)$ the smallest and largest *positive* eigenvalues of M, respectively. We denote the $s-$dimensional identity matrix as I_s and $M^\dagger$ denotes the *pseudo inverse* of matrix M. $\|v\|_M$ denotes the $M-$matrix norm of a vector and is defined as: $\|v\|_M = \sqrt{v^T M v}$.

4.3 Preliminaries & Model

We start by re-introducing the stochastic linear bandits framework and meta linear bandits for completeness. The notation introduced will be used throughout the paper.

4.3.1 Stochastic Linear Bandits

The standard linear bandit problem is the problem that a set of $d-$dimensional arms are revealed to the agent, where the agent's goal is to maximize the expected cumulative reward in this $T-$round online game. The agent at each $t \in [T]$ tries to estimate a parameter $\theta^* \in \mathbb{R}^d$ as accurate as possible by immediately observing the rewards. Mean rewards are bilinear in feature vectors with additive sub-Gaussian noise, i.e.

$$r(a) = \langle a, \theta \rangle + \eta \tag{4.1}$$

Assumption 4.3.1. The following are the common assumptions in literature.

1. Bounded arms: $\|a\| \leqslant L$ for all $a \in A$
2. Bounded θ: $\|\theta^*\| \leqslant S$
3. Sub-Gaussian noise: $\|\eta\|_{\psi_2} \leqslant \sigma$
4. Finite arm set: $A = \{a_1, \ldots, a_K\}$
5. Bounded rewards: $\langle a, \theta^* \rangle \in [-1, 1]$

The goal of the agent is to maximize cumulative reward, or better yet, minimize a metric called *regret*. Let r_t denote the immediate regret given as

$$r_t = \langle a^* - a_t, \theta^* \rangle \tag{4.2}$$

where a^* is the arm that maximizes the expected reward, formally

$$a^* = argmax_{a \in A} \langle a, \theta^* \rangle \tag{4.3}$$

LinUCB Algorithm The standard stochastic linear bandit problem has been studied extensively, and has various order optimal solutions up-to logarithmic factors. A well-known algorithm is called LinUCB and/or OFUL in different works [2] , [25]. Below we give an overview of the practically identical methodology studied in these, and what our algorithm will eventually largely draw from.

Confidence Sets The LinUCB algorithm relies on using previously-observed rewards to construct carefully tuned confidence sets on θ^*. The following construction is based on that of [1]: Let $\lambda > 0$ be an arbitrary constant (in practice, this can be treated as a tuning parameter). Suppose at the beginning of time $t + 1$, the policy at hand has selected arms $a_1, \ldots, a_t$ and observed rewards $r_1, \ldots, r_t$. We construct the matrix Σ_t as

$$\Sigma_t = \sum_{i=1}^{t} a_i {a_i}^\top + \lambda I,$$

where I is the identity matrix. Then letting $\hat{\theta}_t$ denote the regularized least-squares estimator of θ^*, we have

$$\hat{\theta}_t \equiv \Sigma_t^{-1} \sum_{i=1}^{t} a_i r_i,$$

and finally, the confidence set used in the LinUCB algorithm is the following ellipsoid:

$$C_t = \{\theta \in \mathbb{R}^d : \|\theta - \hat{\theta}_t\|_{\Sigma_{t-1}} \leqslant \beta_t\}, \tag{4.4}$$

where

$$\beta_t = \sigma \sqrt{2 \log T + d \log \frac{d\lambda + TL^2}{d\lambda}} + \sqrt{\lambda} S. \tag{4.5}$$

The LinUCB algorithm, then, selects the arm that has the highest "potential" reward over all $\theta \in C_t$, i.e. it selects

$$\underset{a \in A}{\text{argmax}} \max_{\theta \in C_t} \langle a, \theta \rangle.$$

The ellipsoidal form of C_t enables a closed-form expression for the inner maximization above:

$$UCB_t(a) \equiv \max_{\theta \in C_t} \langle a, \theta \rangle = \langle a, \hat{\theta}_t \rangle + \beta_t \|a\|_{V_t^{-1}}. \tag{4.6}$$

The algorithm has a regret upper bound of $\tilde{O}(d\sqrt{T})$, where $\tilde{O}$ hides logarithmic terms. LinUCB is order optimal, since the lower bound on regret of stochastic linear bandits is $\Omega(d\sqrt{T})$.

4.3.2 Meta Linear Bandits

Meta linear bandits are a sequence of linear bandit games with a finite horizon. Each of these games are called a *period*. Let N denote the number of distinct users arriving at the system. At a period $j = 1, \ldots, N$, and time $t = 1, \ldots, T$, a user arrives with an unknown feature vector $\theta^j \in \mathbb{R}^d$. The reward function follows the linear function introduced in Equation (4.1).

The meta structure behind this model is that these vectors $\theta_1, \ldots, \theta_N$ come from an $s-$dimensional subspace where $s \ll d$. In other words, there exists a $V \in \mathbb{R}^{d \times s}$, such that $VV^\top \theta^j = \theta^j$ for all $j \in [N]$. Meta linear bandits showcase a regret lower bound of $\Omega(d + \sqrt{sdNT} + Ns\sqrt{T})$. There exists a concept of *oracle* in meta linear bandits, where the algorithm knows the matrix V. In order to evade the high dimensional curse, the oracle projects arms onto the $s-$dimensional subspace spanned by the vectors of V, and executes an order optimal algorithm. One such algorithm with optimism principle as introduced in the previous section, i.e. LinUCB, is given in Algorithm 3 for an episode.

Algorithm 3: LinUCB-s

Input: V_s, A, c
Initialization: $V^j = \lambda I$, $\hat{\theta}_j = [0]^s$
Project arms onto $s-$dimensional subspace: $\tilde{A} := \{V_s a : a \in A\}$
for $t = 1, 2, \ldots T$ **do**
  Select arm $a_{LinUCBs_t} = \text{argmax}_{a \in \tilde{A}}\, a^\top \hat{\theta}^{j_t} + c\|a\|_{\tilde{V}_{t-1}^{-1}}$
  Observe reward r and append to r^{j_t}
  Update $V^{j_t} = V^{j_t} + a_{LinUCBs_t}(a_{LinUCBs_t})^\top$
  Update $\hat{\theta}$
end

The algorithm's working principle feeds from the following fact where $P = V^\top V$:

$$a^\top \theta = a^\top P \theta = (Va)^\top V\theta$$

Hence the algorithm learns only on the component that is spanned by V, since the space spanned by $V^\perp$ does not contribute to the reward.

4.3.3 Main Model: Bilinear Bandits with Arbitrary Arrivals

We herein introduce a more general version of meta linear bandits, where there are multiple users arriving in the system *in an arbitrary fashion*. Formally, let $\theta_j^* \in \mathbb{R}^d$ denote the true latent vector of a user j, where there are N users in total: $j = 1, \ldots, N$. The reward function follows the linear function introduced in Equation (4.1) for each user and arm pulled. The game consists of a single T-length game, where at each time t the system receives an index of the user, pulls an arm a_t and receives reward r_t. The reward function follows the linear function with sub-Gaussian noise as usual, as defined in Equation (4.1).

As per the meta regime, we assume that these vectors lie in a lower $s-$dimensional linear subspace such that $s \ll d$. In other words, they can be written in the form $\theta = \sum_{i=1}^{s} 1_i v_i$ for some $1_1, \ldots, 1_s \in \mathbb{R}$, and $v_i \perp v_j$ for all $i, j \in [s]$. For the model to be meaningful we also require that during the horizon T in consideration, we observe at least $\Omega(s)$ arrivals to the system for each user j. At a given time $t \in \{1, \ldots, T\}$, a game/parameter $j_t \in [N]$ is received with its unknown $\theta_{j_t}^* \in \mathbb{R}^d$. Notice that the order $j_1, \ldots, j_T$ can be arbitrary, and the only assumption we make is each use arrives at least s times, meaning, $\sum_t 1(j_t = j) \geqslant s$ for all $j \in [N]$. This is a necessary assumption since, even when the underlying subspace is known, in the case for $s \gg T^j$, no algorithm can do better than random. In other words, any algorithm needs at least s observations to form an $s-$dimensional estimate, hence, $s \ll T^j$ will result in linear regret. At each round t, with the current user θ_{j_t}, a decision is made so as to pick an action $a \in \mathbb{R}^d$ from the set A, where $|A| = K$. Notice that the arm set could also have a dynamic structure throughout time, and be dependant on t, however, for notation ease we work with fixed set of arms. The work is easily extendable to varying arm sets over time.

Metric The metric we try to maximize in this game is *cumulative reward*. Notice that the optimal policy with the full knowledge of a_i and θ_j would be to maximize the expected cumulative reward. Let π denote any *policy*,

$$\pi_t : ([K] \times \mathbb{R} \times [N])^{t-1} \times [N] \to [K],$$

which at each time t, maps the previously selected arms and observed rewards to the next selected arm. Furthermore, let a_{π_t} be the arm that is played by the algorithm at that time. Now, for any given policy π, we try to maximize the expected cumulative reward given as:

$$\sum_t r(a_{\pi_t}) = \sum_t \langle \theta^*_{j_t}, a_{\pi_t} \rangle \tag{4.7}$$

For the purpose of experiments and algorithms, we will compare the results based on this given cumulative reward metric.

4.4 Our Algorithm

Before diving into the algorithm proposed, the issue we address should be made clear. In the high dimensional case we are interested in, $d \gg T^j$ for all $j \in [N]$, for an algorithm that works in $d-$dimensions, there is no accurate estimate of each θ^j arriving in the system as the well-known estimation methods require a certain number of observations in $O(d)$. Stemming from this issue faced in real-life recommender systems, the general idea behind our algorithm is to operate on an approximated lower dimensional setting so as to avoid the drawbacks from the high dimensional setting.

More rigorously, $X^n_t \in \mathbb{R}^{d \times t}$ denotes the matrix whose columns are the arms played up until time t in user type n. Σ^n_t is the regularized matrix in the form $\Sigma^n_t = \lambda I + X^n_t (X^n_t)^\top$ where λ is a regularization parameter. The main method is that after each episode ends, we re-calculate the matrix $\hat{V}$ using the previously observed data and calculated $\hat{\theta}$. Note that at the beginning of the game, since $T \ll d$, we do require a certain number of episodes, $\tilde{n}$, dedicated to exploration. During these episodes, the algorithm picks arms orthogonal to each other so as to maximize the information gain from them. Namely, let $X^n \in \mathbb{R}^{T \times d}$ be the arm

matrix whose rows are the arms played up until the end at episode n.The algorithm then is to pick arms at each episode n such that the arms played are unit vectors of $d-$dimensional space: $e_i \in \mathbb{R}^d$ for all $i \in [d]$.

First, for each of these θ^n, we create a $d-$dimensional Ridge estimator $\hat{\theta}^n$ such that:

$$\hat{\theta}_T^n = (\Sigma_T^n)^{-1}(X_T^n)^\top Y_T^n$$

where $Y_T^n \in \mathbb{R}^T$ is the noisy reward observed throughout episode n for T rounds.

Below in Algorithm 3, we provide LinUCB algorithm on a lower dimensional setting, where we require a V_s-matrix to switch to a lower dimensional setting. The algorithm has the same logic explained in Section 4.3.1, except that it projects the high dimensional arms onto the given subspace at the beginning of the game using matrix V_s, and works with the projected armset $\tilde{A}$ throughout T rounds. For each $t = 1, \ldots, T = d$ we try to solve for the following problem:

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & rank(\Theta) \\ \text{subject to} \quad & \sum_{n=1}^{N} \|\bar{A}_n \Theta_n - r_n\|^2 \leqslant \gamma \end{aligned}$$

which we relax as:

$$\underset{\Theta}{\text{minimize}} \quad \frac{1}{2} \sum_{n=1}^{N} \|\bar{A}_n \Theta_n - r_n\|^2 + \lambda \|\Theta\|_*$$

The Iterative Algorithm below solves this relaxed version of the problem approximately.

Algorithm 4: Iterative Algorithm

Input: $\Theta^{i-1}, r, \bar{A}, \lambda$
for $n = 1, \ldots, N$ **do**
  Solve for $\theta_n^* = \text{argmin}_{\theta_n} \|\theta_n - \Theta_n^{i-1}\|^2 + \gamma \|\bar{A}_n \theta_n - r_n\|^2$
end
Construct $\Theta^{int} = [\theta_1^*, \ldots, \theta_N^*]$
Calculate $U^{int}, \Sigma^{int}, V^{int} = SVD(\Theta^{int})$
Return $\Theta^i = U^{int} \Sigma_\lambda^{int} V^{int}$ and V^{int} where Σ_λ is the soft threshold.

The idea behind the iterative algorithm is as follows. For a very large d, in a single bandit game, for the first d rounds the algorithm cannot do anything but random. However, in the case we know the underlying dimension among these unknown θs, we can come up with

a constrained optimization problem where we minimize the rank of the design matrix of incoming user feature vectors. Then, by penalizing the difference between the estimated rewards of current $\hat{\Theta}$ and given arms, we estimate *each* θ_j that arrived in the sysxtem until time t. Then we perform a SoftImpute based SVD, where SoftImpute is the algorithm given in Algorithm 6. Then, we repeat this approximate rank minimization and SoftImpute until convergence.

As per what happens when a user arrives, as given in Line 5, the estimated subspace for θ values are used to project arms onto the space. Then, a regular LinUCB is performed, where the only difference is the $\hat{\theta}$ is *not* the maximum likelihood estimator, but the estimate we acquired from the iterative approximation of the matrix Θ.

Optimism Principle Having mentioned the algorithm we use to estimate a $\hat{\theta}_j$ for each user, we further want to improve by introducing an optimism based principle. As given in Algorithm 3, LinUCB introduces a certain exploration based on confidence intervals obtained on θ. With that idea in mind, hereby, we borrow from a very recent work on the entry wise confidence intervals on matrix completion [29].

Algorithm 5: High Dimensional Bilinear Low Rank LinUCB

[h!]
Input: c, λ
Initialize: $\hat{\theta}_0^1, \ldots, \hat{\theta}_0^N = [0]^d, V_0^n = \lambda I, \Theta^0 = [0]^{N \times d}$
for *t=1,. . . ,T* **do**
  $\Theta_t, X_t = \text{IterativeAlgorithm}(\Theta_{t-1}, r, \bar{A}, \lambda)$
  $[\hat{\theta}_{t-1}^1, \ldots, \hat{\theta}_{t-1}^N] \leftarrow \Theta_t$
  Project arms onto estimated subspace $\tilde{A}_t^n = X_t A^n$
  Update/calculate V_{t-1}^n according to this projection matrix/projected arm set
  Play arm $\text{argmax}_{a \in \tilde{A}_t^n} a^\top \hat{\theta}_{t-1}^n + c s_{j_t,a}$ where s_{ij} is as defined in Equation (4.8) and observe reward
  Update r and $\bar{A}$
end

Lemma 4.4.1 (Farias et al, [29]). *For binary observations, the entry* (i, j) *has a variance*

$$s_{ij}^2 = \frac{\sum_{l=1}^m M_{lj}^*(1 - M_{lj}^*)(\sum_{k=1}^r U_{ik}^* U_{lk}^*)^2}{p} + \frac{\sum_{l=1}^n M_{il}^*(1 - M_{il}^*)(\sum_{k=1}^r V_{lk}^* V_{jk}^*)^2}{p} \tag{4.8}$$

The full form of the theorem can be found in [29]. There are two issues with the practical correspondence of this theoretical breakthrough: the model this theorem is based on assumes that entries are observed independently with a probability p *and* it assumes the arms(or column vectors) are unit vectors. We conjecture the second assumption could be relaxed to fit this work, however, the theoretical foundation of the algorithm introduced is beyond the scope of this work. We nevertheless apply this theorem to form a confidence interval on the entries. The corresponding algorithm is given in Algorithm 5. The working logic is as follows: we first solve the rank minimization problem by relaxing and applying SoftImpute. This is not a new development, since it is the foundational method for matrix completion algorithms. Then, with the calculated θs, we create a confidence interval for each entry in the reward matrix using Equation (4.8), and pick the entry that gives the highest potential reward. The idea is to introduce a certain optimism principle based on the entry-wise variance where the mean is the estimated θ values.

4.5 Experiments

In this section, we work on a dataset that was acquired from [81]. We first give an analsyis of the dataset, then the detailed explanation of how we retrieved a ground truth for comparison and a set of algorithms from literature that we compare against.

4.5.1 Data Description

In this section, we introduce and analyze a dataset from NetEase, a music streaming platform. The detailed full description of this set can be found in [81]. NCM is a free music streaming service developed and owned by NetEase, Inc. For the purposes of this experiment, we work on the impression dataset. Table 4.1 describes each data field in the impression-level data table. This data table contains $57,750,395$ impression-level data points covering $2,085,533$ unique users during the 30-day-long sample period.

The table contains 13 data fields. The userId data field uniquely identifies each user in the entire data set, and it can be used to join this table with user tables in Section 2.4. The mlogId data field uniquely identifies each card and can be used to join this impression

Column Name	Data Type	Description
userId	string	The unique identifier of each user in the data set
dt	numeric	The number of days from the start of the sample period
mlogId	string	The unique identifier of each card in the data set.
impressTime	numeric	The epoch time of the impression
impressPosition	numeric	The position of impression in the feed
isClick	binary	1 if the user clicks on the card, 0 ow 1
isComment	binary	1 if the user comments on the card, 0 ow
isLike	binary	1 if the user likes the card, 0 ow 1
isShare	binary	1 if the user shares the card, 0 ow 1
isViewComment	binary	1 if the user views comments from other users on the card, 0 ow
isIntoPersonalHomepage	binary	1 if the user enters the creator's homepage through the card, 0 ow
mlogViewTime	numeric	The number of seconds that the user has spent on the card
detailMlogInfoList	string	JSON file contains all cards that the user sees if s/he swipes down

Table 4.1: Table for all the columns the impression data contains

table with card tables in Section 2.2. The impressTime data field records the epoch time when the impression is first shown to the user on her app. Each user may have multiple impressions in a given day; each card may be shown to multiple users during the sample period. Therefore, each row of impression data is uniquely identified by a combination of userId, mlogId and impressTime, representing that a card is shown to a user at a particular time. Each impression for a user comes with a position in her feed stream, and it is recorded in the impressPosition data field. The position starts with 1 and is counted from top to down and from left to right.

For each impression, the table provides the users' actions on the recommended cards. First, a user could click on a card once the impression of the card is presented to the user. Once the user clicks on the card, the music video in the card will be automatically played in the user's app in full screen mode if the card contains a video. If a card contains a set of images, the first image will be shown to the user in full screen mode. Such an action is recorded in the isClick data field. Second,once clicking on a card, the user can comment and view other comments on the card. Whether a user comments on a card for an impression is recorded in the isComment data field, and whether a user views others' comments on a recommended card is recorded in the isViewComment data field.

A user can also like a shown card. Whether a user likes a card or not is recorded in the

isLike data field. A user can also click on the share button. Whether a user decides to share the card from an impression is recorded in the isShare data field. Moreover, a user can click on the creator's personal profile logo on top of the like button and get into the creator's personal page, and whether a user enters the creator's personal homepage from a card is recorded in the isIntoPersonalHomepage data field. Last but not least, the users can also swipe down a card. Once a user decides to swipe down the card, a new card will be automatically recommended to the user. If a user chooses to swipe down after watching a card recommended to her, the information regarding each of the cards shown after swiping down will be stored in the detailMlogInfoList data field of the focal card's data point in Table 2. The number of data points in detailMlogInfoList represents how many times that the user has swiped down after clicking on a card. Furthermore, the data also provides the number of seconds for which the user has played the card. In other words, this is the difference between the time when the user clicks on the card and the time when the user swipes down or leaves the card by clicking the back button or closes the app. If the card contains a video and the user is still on the video page when the video ends, the video will automatically replay. The watch time of an impression is recorded in the mlogViewTime data field. Notice that a user's total app usage cannot be imputed from this watch time since a user may browse other tabs in the app on a given day; and therefore, unfortunately, researchers cannot impute a user's total app usage in a day through this data set.

4.5.2 Data Analysis

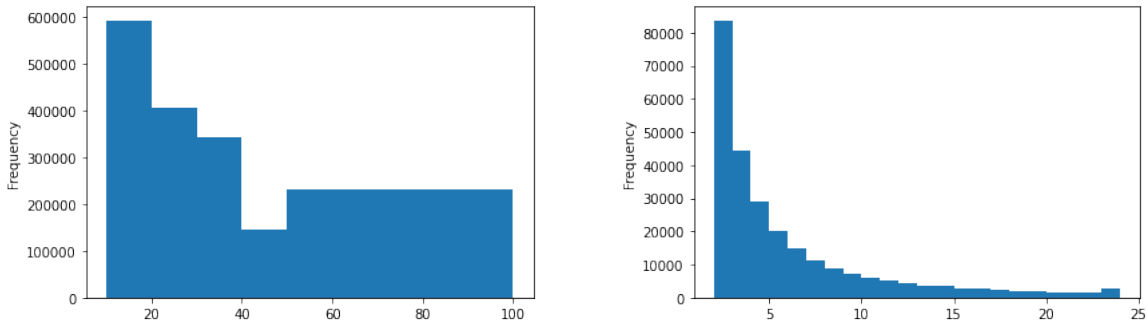
In this section we do a preliminary statistical analysis of the data in hand.

	isClick	isComment	isLike	isShare
Positive	427524	8629	141210	19518
Negative	54879957	57741766	57609185	57730877

Table 4.2: The positive and negative feedback amount based on the type of feedback the system received. Note that negative feedback is that there is no interaction of that type for a given impression point.

We further provide histogram for the amount of time spent in a given recommendation. Notice that the majority of the users spend less than 20 minutes on a given recommendation.

Furthermore, most recommended cards are not interacted at all, if one looks at Table 4.2.



(a) Number of impressions vs time spent

(b) User number vs number of impressions

Figure 4-1: A snippet of how user behavior is distributed

The Ground Truth We make the following assumptions before implementing an algorithm to extract a pseudo *ground truth*.

1. Rewards are binary: $r \in \{0, 1\}$.
2. The reward takes value 1, if at least 1 of the isView, isComment, isClick,isLike,isShare fields is 1.
3. The reward matrix is low rank.

Under the light of the assumptions above, we employ one of the commonly used matrix completion algorithms to extract the true feature vectors of users and items, called SoftImpute. We take the following notation and algorithm from [59].

Define a matrix $P_\Omega(Y) \in \mathbb{R}^{m \times n}$ such that

$$P_\Omega(Y)(i,j) = \begin{cases} Y_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{if } (i,j) \notin \Omega \end{cases} \tag{4.9}$$

Lemma 4.5.1 (Mazumder et al. , [59])**.** *Suppose the matrix $W_{m \times n}$ has rank r. Then the solution to the optimization problem*

$$\min_Z \frac{1}{2}\|W - Z\|_F^2 + \lambda \|Z\|_* \tag{4.10}$$

is given by $\hat{Z} = S_\lambda(W)$ *where*

$$S_\lambda(W) \equiv UD_\lambda V' \tag{4.11}$$

with $D_\lambda = diag[(d_1 - \lambda)_+, \ldots, (d_r - \lambda)_+]$. UDV' *is the SVD of* W, $D = diag[d_1, \ldots, d_r]$ *and* $t_+ = \max(t, 0)$.

In the light of the lemma, we also employ the following algorithm to extract the ground truth.

Algorithm 6: Soft Impute

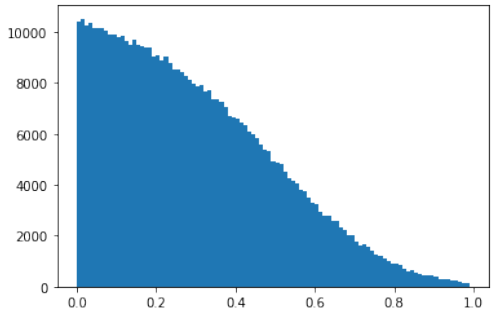
Initialize: $Z^{old} = 0$
for $\lambda_1 > \cdots > \lambda_r$ **do**
  while *True* **do**
    Repeat:
    Compute $Z^{new} \leftarrow S_{\lambda_r}(P_\Omega(X) + P_\Omega^\perp(Z^{old}))$
    If $\frac{\|Z^{new} - Z^{old}\|_F^2}{\|Z^{old}\|_F^2} < \epsilon$ then break.
    Assign $Z^{old} \leftarrow Z^{new}$
  end
  Assign $\hat{Z}_{\lambda_r} \leftarrow Z^{new}$
end
Return $\hat{Z}_{\lambda_1}, \ldots, \hat{Z}_{\lambda_r}$.

Ambient and true dimension For the purposes of our experiments, we tested various dimensions/rank while implementing SoftImpute: $d = 20, 200, 2000, 10000$, which denotes the ambient dimension all θ and arm vectors lay in. We further employ SVD methods to perform PCA and project the vectors onto $s-$dimensional subspace. Mainly, let $\theta_1, \ldots, \theta_N \mathbb{R}^d$ denote the extracted user features from data. We re-calculate the $d-$dimensional representations of these vectors by projecting them onto a subspace extracted from $s-$principal components using SVD. Namely,

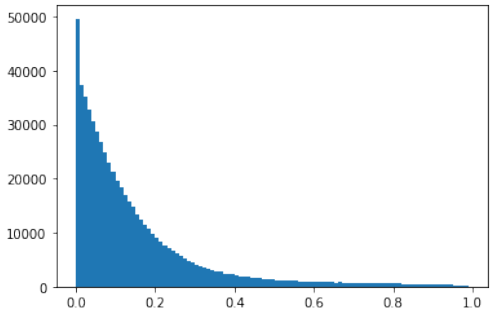
$$\Theta = U\Sigma V^\top \tag{4.12}$$

Then we take the first s dimensions of this decomposition.

$$\Theta' = U\Sigma_s V^\top \tag{4.13}$$



(a) Mean rewards histogram for d=20



(b) Mean rewards histogram for d=200

where $\Sigma_s = \begin{bmatrix} s_1 & & 0 & \dots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & s_r & & 0 \\ \vdots & & & \ddots & \\ 0 & & 0 & & 0 \end{bmatrix}$. Then we multiple each θ' with V, to get a $d-$dimensional representation with an underlying $s-$dimensional subspace. Notice that there are multiple ways to extract such information, another one of which is to randomly generate a $d-$dimensional spanning vector set and represent these newly found $s-$dimensional user vectors by multiplying with these vectors.

Below, we give the reward distribution acquired after implementing SoftImpute matrix completion method for $d = 20$ and $d = 200$. Notice that mean rewards change according to the dimension we use to complete the rewards matrix from raw data.

4.5.3 Benchmark Algorithms

Matrix Completion and Entry-wise Confidence Intervals In terms of matrix completion and the entry-wise confidence intervals, there haven't been a development until recently. We adapt theoretical results from one of the aforementioned papers [29].

Under the assumptions, an entry M_{ij} follows a distribution with the following variance:

$$s_{ij}^2 = \frac{\sum_{l=1}^{m} M_{lj}^*(1 - M_{lj}^*)(\sum_{k=1}^{r} U_{ik}^* U_{lk}^*)^2}{p} + \frac{\sum_{l=1}^{n} M_{il}^*(1 - M_{il}^*)(\sum_{k=1}^{r} V_{lk}^* V_{jk}^*)^2}{p} \tag{4.14}$$

Notice that we do not have access to true reward matrix, user matrix, and arm matrix: M^*, U^*, V^* respectively. Hence, we instead use the estimated versions of these matrices

Arm Information	Greedy	Optimism
s-dimensional arms	-	oracle
d-dimensional arms with rank info s	-	ours
d-dimensional arms	-	Separate
no arm information with rank info s	MC-Greedy	MC

Table 4.3: The algorithms categories based on given information

extracted by SoftImpute algorithm. We test our algorithm introduced in the previous section against the following benchmarks. The following table demonstrates the algorithms we test with varying information regimes.

1. Separate: The algorithm we call 'Separate' has no information about the underlying subspace, or that there exists one. It treats each distinct user arrival independently and plays a linear bandit game in parallel with N users.

2. Oracle: This is the alogrithm that has all the information except true θs.

3. MC + UCB: This algorithm makes use of the fact that the reward matrix is low rank and applies the SoftImpute algorithm to extract an s-dimensional estimate of both sets of vectors Θ, A.

4. MC + Greedy: This is the algorithm that executes a matrix completion algorithm with $s-$dimensional rank. Then, it picks a largest estimated value among the values in the current user in consideration.

5. Ours: We apply the algorithm given in the previous section, where we utilize the low-rank knowledge on arriving users, estimate the parameters and add a exploration term based on entry based variances.

4.5.4 Results

After running the algorithms described in the previous subsection for the first 1 million arrivals, and cumulatively calculating the reward we obtain the following results. The experiments were repeated over 100 times. Note that the reward we care about is the mean rewards rather than the actual observed ones, since that is what we want to maximize for.

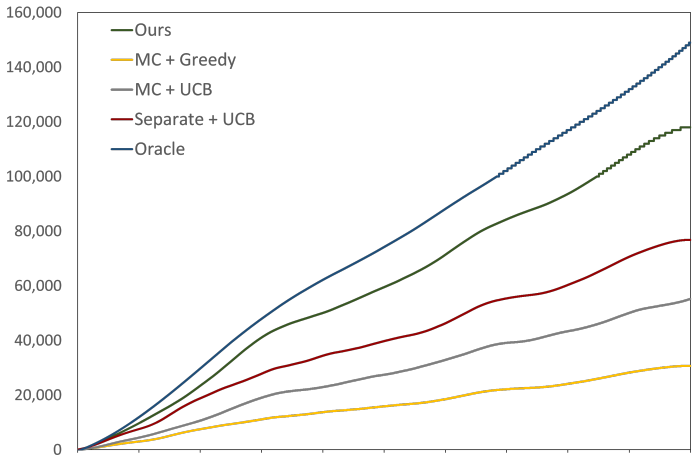
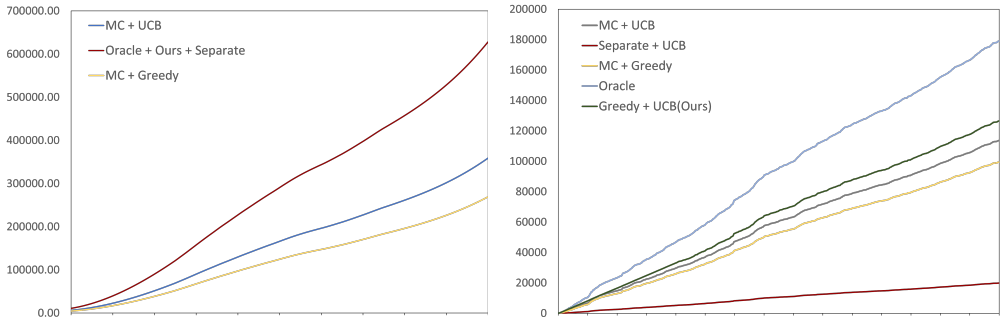


Figure 4-2: Reward vs Time plot for d=200, s=20

We further look at the effect of the ratio d/s on the results, and run the experiments for when $d = s = 20$ and $d = 2000, s = 20$. Notice that for the first case, our algorithm, the oracle, and Separate become the same as the ambient dimension and the true dimension coincide.



(a) Reward vs Arrival for d=20 and s=20 **(b)** Reward vs Arrival for d=2000 and s=20

Figure 4-3: Extreme cases of dimensions

Analysis of results As seen in Figure 4-2, we have established that for $s = 20, d = 200$ the algorithm proposed has better performance than the aforementioned algorithm adaptations in literature. We further investigate the effect of dimension ratio i.e. d/s on the algorithm performance. We point out that the case for which the underlying dimension is equal to

that of the introduced dimension, i.e. $d = s = 20$, the algorithms *Separate, Oracle, Ours* become the same, since the information of the underlying dimension does not constitute for additional information. We see that in that case MC methods work well enough, within 87% of the oracle's cumulative reward.

For these 2 experiments, our algorithm has made use of the knowledge of arms and won against the other benchmarks. We further question the effect of high dimension by working with $d = 2000$. In Figure 4-2, it is clear that the knowledge on the arms cannot overcome the low rank assumption as much when we work with higher dimensional settings. There are 2 reasons for this. The first one is, for 10 000 users and such short arrival time, any estimation for $d = 2000$ is going to be very close to random, which is what causes the Separate method to fail in the higher dimensional setting. Another point is, tour algorithm becomes very close to those based on matrix completion methods. This is because the knowledge on the arms do not matter as much. Or in other words, the matrix completion methods are doing a good enough job to compensate for the lack of knowledge in 2000−dimensional arms.

4.6 Conclusion

In this paper, we investigate the general version of high dimensional meta linear bandits, where there is an underlying lower dimensional subspace the unknown user feature vectors come from. The users arrive in the system in an arbitrary fashion, and we assume each user arrives at least m times through the horizon T. We introduce a bandit model where rewards are binary. We further assume the mean of these rewards are bilinear in the user vectors. We introduce an algorithm to solve this problem where we solve a rank minimization problem subject to certain constraints gathered from past data. We validate the proposed algorithm against various other algorithms from the literature on a real world data-set, and see the algorithm outperforms the well known algorithms in literature. We further investigate the effect of the relationship between the true and ambient dimension and how it affects the final outcome of these algorithms.

Future work includes theoretical bounds for this algorithm, and further investigation on different data-sets and how the distribution of items and users might affect the final outcome.

Bibliography

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[2] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

[3] Shipra Agrawal and Nikhil Devanur. Linear contextual bandits with knapsacks. In *Advances in Neural Information Processing Systems*, pages 3450–3458, 2016.

[4] Shipra Agrawal and Nikhil R. Devanur. Bandits with global convex constraints and objective. *Operations Research*, 67(5):1486–1502, 2019.

[5] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 127–135, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[6] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[7] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216. IEEE, 2013.

[8] Hamsa Bastani, David Simchi-Levi, and Ruihao Zhu. Meta dynamic pricing: Transfer learning across experiments. *Management Science*, 2021.

[9] Soumya Basu, Orestis Papadigenopoulos, Constantine Caramanis, and Sanjay Shakkottai. Contextual blocking bandits. *arXiv preprint arXiv:2003.03426*, 2020.

[10] Soumya Basu, Rajat Sen, Sujay Sanghavi, and Sanjay Shakkottai. Blocking bandits. *CoRR*, abs/1907.11975, 2019.

[11] Dimitris Bertsimas and Adam J Mersereau. A learning approach for interactive marketing to a customer segmenta. *Operations Research*, 55(6):1120–1135, 2007.

[12] Dimitris Bertsimas and José Niño-Mora. Restless bandits, linear programming relaxations, and a primal-dual index heuristic. *Operations Research*, 48(1):80–90, 2000.

[13] Daniel Billsus and Michael J Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3):147–180, 2000.

[14] JesúS Bobadilla, Fernando Ortega, Antonio Hernando, and JesúS Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-based systems*, 26:225–238, 2012.

[15] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[16] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Gançarski. A contextual-bandit algorithm for mobile context-aware recommender system. In Tingwen Huang, Zhigang Zeng, Chuandong Li, and Chi Sing Leung, editors, *Neural Information Processing*, pages 324–331, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[17] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

[18] Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[19] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[20] Felipe Caro and Jérémie Gallien. Dynamic assortment with demand learning for seasonal consumer goods. *Management Science*, 53(2):276–292, 2007.

[21] Leonardo Cella, Alessandro Lazaric, and Massimiliano Pontil. Meta-learning with stochastic linear bandits. In *International Conference on Machine Learning*, pages 1360–1370. PMLR, 2020.

[22] Nicolò Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. A gang of bandits. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 737–745, Red Hook, NY, USA, 2013. Curran Associates Inc.

[23] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. pages 273–280, 2009.

[24] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 208–214, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[25] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[26] Richard Combes, Chong Jiang, and Rayadurgam Srikant. Bandits with budgets: Regret lower bounds and optimal algorithms. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):245–257, 2015.

[27] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[28] Aniket Anand Deshmukh, Urun Dogan, and Clayton Scott. Multi-task learning for contextual bandits, 2017.

[29] Vivek F. Farias, Andrew A. Li, and Tianyi Peng. Uncertainty quantification for low-rank matrix completion with heterogeneous and sub-exponential noise, 2021.

[30] Vivek F. Farias and Ritesh Madan. The irrevocable multiarmed bandit problem. *Oper. Res.*, 59(2):383–399, March 2011.

[31] Chao Gan, Ruida Zhou, Jing Yang, and Cong Shen. Cost-aware cascading bandits. *IEEE Transactions on Signal Processing*, 68:3692–3706, 2020.

[32] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Evans Etrue, and Giovanni Zappella. On context-dependent clustering of bandits, 2016.

[33] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits, 2014.

[34] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. A scalable, accurate hybrid recommender system. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 94–98. IEEE, 2010.

[35] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[36] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, and Francisco Carrero García. Content based sms spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering*, pages 107–114. ACM, 2006.

[37] Aditya Gopalan, Odalric-Ambrym Maillard, and Mohammadi Zaki. Low-rank bandits with latent mixtures, 2016.

[38] Aditya Gopalan, Odalric-Ambrym Maillard, and Mohammadi Zaki. Low-rank bandits with latent mixtures, 2016.

[39] Botao Hao, Tor Lattimore, and Mengdi Wang. High-dimensional sparse linear bandits. *Advances in Neural Information Processing Systems*, 33:10753–10763, 2020.

[40] Nicholas Johnson, Vidyashankar Sivakumar, and Arindam Banerjee. Structured stochastic linear bandits, 2016.

[41] Kwang-Sung Jun, Rebecca Willett, Stephen Wright, and Robert Nowak. Bilinear bandits with low-rank structure, 2019.

[42] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, Claire Vernade, and Zheng Wen. Stochastic rank-1 bandits, 2016.

[43] Jaya Kawale, Hung Bui, Branislav Kveton, Long Tran Thanh, and Sanjay Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1297–1305, Cambridge, MA, USA, 2015. MIT Press.

[44] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE transactions on information theory*, 56(6):2980–2998, 2010.

[45] Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

[46] TomÃ¡Å¡ KocÃ¡k, RÃ©mi Munos, Branislav Kveton, Shipra Agrawal, and Michal Valko. Spectral bandits. *Journal of Machine Learning Research*, 21(218):1–44, 2020.

[47] Branislav Kveton, Csaba Szepesvari, Anup Rao, Zheng Wen, Yasin Abbasi-Yadkori, and S. Muthukrishnan. Stochastic low-rank bandits, 2017.

[48] T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4 – 22, 1985.

[49] Sahin Lale, Kamyar Azizzadenesheli, Anima Anandkumar, and Babak Hassibi. Stochastic linear bandits with hidden low rank structure, 2019.

[50] Sahin Lale, Kamyar Azizzadenesheli, Anima Anandkumar, and Babak Hassibi. Stochastic linear bandits with hidden low rank structure, 2019.

[51] Ken Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95*, 1995.

[52] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[53] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[54] N. Levine, K. Crammer, and Shie Mannor. Rotting bandits. In *NIPS*, 2017.

[55] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA, 2010. ACM.

[56] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[57] Yangyi Lu, Amirhossein Meisami, and Ambuj Tewari. Low-rank generalized linear bandit problems, 2020.

[58] Yangyi Lu, Amirhossein Meisami, and Ambuj Tewari. Low-rank generalized linear bandit problems, 2020.

[59] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

[60] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, 23:187–192, 2002.

[61] Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000.

[62] Andy Ramlatchan, Mengyun Yang, Quan Liu, Min Li, Jianxin Wang, and Yaohang Li. A survey of matrix completion methods for recommendation systems. *Big Data Mining and Analytics*, 1(4):308–323, 2018.

[63] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12), 2011.

[64] Herbert Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58(5):527–535, 09 1952.

[65] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295, 2001.

[66] Ingo Schwab, Alfred Kobsa, and Ivan Koychev. Learning user interests through positive examples using content analysis and collaborative filtering. In *30 2001. Internal Memo, GMD*, 2001.

[67] Rajat Sen, Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G. Dimakis, and Sanjay Shakkottai. Contextual bandits with latent confounders: An nmf approach, 2016.

[68] Julien Seznec, Andrea Locatelli, Alexandra Carpentier, Alessandro Lazaric, and Michal Valko. Rotting bandits are no harder than stochastic ones. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2564–2572. PMLR, 16–18 Apr 2019.

[69] Nícollas Silva, Heitor Werneck, Thiago Silva, Adriano CM Pereira, and Leonardo Rocha. Multi-armed bandits in recommendation systems: A survey of the state-of-the-art and future directions. *Expert Systems with Applications*, 197:116669, 2022.

[70] Aleksandrs Slivkins. Dynamic ad allocation: Bandits with budgets. *ArXiv*, abs/1306.0155, 2013.

[71] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, Aug 2011.

[72] Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008.

[73] Sharan Vaswani, Mark Schmidt, and Laks V. S. Lakshmanan. Horde of bandits using gaussian markov random fields, 2017.

[74] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, 69:29–39, 2017.

[75] Yingce Xia, Haifang Li, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Thompson sampling for budgeted multi-armed bandits. *arXiv preprint arXiv:1505.00146*, 2015.

[76] Yingce Xia, Tao Qin, Wenkui Ding, Haifang Li, Xudong Zhang, Nenghai Yu, and Tie-Yan Liu. Finite budget analysis of multi-armed bandit problems. *Neurocomputing*, 258:13 – 29, 2017. Special Issue on Machine Learning.

[77] Miao Xie, Wotao Yin, and Huan Xu. Autobandit: A meta bandit online learning system. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 5028–5031. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Demo Track.

[78] Y. YU, T. WANG, and R. J. SAMWORTH. A useful variant of the davis—kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

[79] Yisong Yue, Sue Ann Hong, and Carlos Guestrin. Hierarchical exploration for accelerating contextual bandits. *arXiv preprint arXiv:1206.6454*, 2012.

[80] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 2025–2034, New York, NY, USA, 2016. ACM.

[81] Dennis J Zhang, Ming Hu, Xiaofei Liu, Yuxiang Wu, and Yong Li. Netease cloud music data. *Manufacturing & Service Operations Management*, 24(1):275–284, 2022.

[82] Jie Zhou, Botao Hao, Zheng Wen, Jingfei Zhang, and Will Wei Sun. Stochastic low-rank tensor bandits for multi-dimensional online decision making, 2020.

[83] Shengli Zhu, Jakob Coles, and Sihong Xie. Active search using meta-bandits. In *Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management*, CIKM '20, page 3493–3496, New York, NY, USA, 2020. Association for Computing Machinery.

[84] Julian Zimmert and Yevgeny Seldin. Factored bandits. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2840–2849, Red Hook, NY, USA, 2018. Curran Associates Inc.